МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Модель та інформаційна технологія аналізу результатів використання бібліотек машинного навчання у реалізації проектів розробки програмних продуктів»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-91 Макаренко Дмитро Валерійович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою «___» грудня 2020 р.

Науковий керівник _____ к.т.н., доц. Парфененко Ю.В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____В. В. Шендрик
«___» _____ 2020 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Макаренко Дмитро Валерійович
(прізвище, ім'я, по батькові)

1 Тема проекту Модель та інформаційна технологія аналізу результатів використання бібліотек машинного навчання у реалізації проектів розробки програмних продуктів
затверджена наказом по університету від « 26 » листопада 2020 р. № 1824-III

2 Термін здачі студентом закінченого проекту « 07 » грудня 2020 р.

3 Вхідні дані до проекту методики бібліотек машинного навчання, дані запитань і відповідей з Stack Overflow.

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, проектування інформаційної системи прийняття рішень, визначення шаблонів помилок та виправлень, розробка інформаційної системи.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, мета дипломного проекту, постановка задачі проекту, дослідження аналогів, проектування інформаційної системи, схема процесів розробки інформаційної системи, реалізація, висновки**.**

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Аналіз та огляд проблеми	01.09.20 – 03.09.20	
2	Ідентифікація ідеї проекту	04.09.20 – 12.09.20	
3	Дослідження методів та підходів для розробки	13.09.20 – 01.10.20	
4	Огляд інструментарію	02.10.20 – 04.10.20	
5	Визначення вимог	05.10.20 – 09.10.20	
6	Створення WBS	10.10.20 – 10.10.20	
7	Підготовка даних	11.10.20 – 21.10.20	
8	Дослідження помилок, виправлень	22.10.20 – 05.11.20	
9	Розробка інформаційної технології	06.11.20 – 25.11.20	
10	Апробація та імплементація	26.11.20 – 29.11.20	
11	Здача документації	30.11.20 – 01.12.20	

Магістрант _____ Макаренко Д.В.

Керівник роботи _____ к.т.н., доц. ПарфененкоЮ.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Модель та інформаційна технологія аналізу результатів використання бібліотек машинного навчання у реалізації проектів розробки програмних продуктів».

Пояснювальна записка складається зі вступу, 5 розділів, висновку, списку використаних джерел із 36 найменувань, додатків. Загальний обсяг роботи − 84 сторінок, у тому числі 67 сторінок основного тексту, 3 сторінки списку використаних джерел, 9 сторінок додатків.

Кваліфікаційну роботу магістра присвячено розробці моделі та інформаційної системи визначення аналізу використання бібліотек машинного навчання у реалізації проектів розробки програмних продуктів. В роботі проведено огляд впливу бібліотек машинного навчання на процеси розробки програмних продуктів. У роботі досліджені та визначені шаблони помилок та виправлень використання бібліотек машинного навчання. У роботі було спроектовано модель та інформаційну технологію з точки зору розробника, побудовано контекстна та концептуальна діаграма, реалізовано алгоритм аналізу використання. Результатом проведеної роботи є модель та інформаційна технологія визначення аналізу використання бібліотек машинного навчання. Практичне значення роботи полягає у розробці інформаційної технології, що дає змогу пришвидшити реалізацію проектів розробки програмних продуктів.

Ключові слова: модель, інформаційна технологія, аналіз, машинне навчання, бібліотеки, програмні продукти.

Зміст

ВСТУП

Машинне навчання стає істотним компонентом сучасної програмної розробки при реалізації програмного продукту використовують високо абстрактні бібліотеки машинного навчання для написання програм машинного навчання в своїх інструментах і додатках. Велика кількість бібліотек машинного навчання загальнодоступна для підтримки зростаючого використання машинного навчання в розробці програмного продукту. Наприклад, деякі з популярних бібліотек – Caffe [1], H2O [2], Keras [3], Mahout [4], MLlib [5], scikit-learn [6], Tensorflow [7], Theano [8], Torch [9] і Weka [10]. Всі ці бібліотеки абстрагують від деталей алгоритмів машинного навчання і надають Application Programming Interfaces (API) [11]. Таким чином, при реалізації програмного продукту можна легко писати складні моделі машинного навчання, використовуючи ці API.

Таким чином, стала актуальною необхідність вивчення проблем, з якими стикаються при застосуванні машинного навчання в розробці програмного продукту. Були вивчені проблеми, з якими стикаються розробники при використанні бібліотек машинного навчання для розробки програмного продукту, характеристики помилок, допущених розробниками, шаблони, яким слідують розробники при виправленні цих помилок, а також запропонували нову модель та інформаційних технологій для автоматичного виявлення неправильного використання API в програмному забезпеченні машинного навчання.

Основні висновки і результати показують, що розробники стикаються з найбільшими труднощами на етапі створення моделей машинного навчання; розробка додатків пов'язана з великою кількістю помилок, патернами помилок, з декількома новими і унікальними патернами поряд з новими проблемами.

Мета роботи – розробка інформаційної технології аналізу результатів використання бібліотек машинного навчання у реалізації проектів розробки програмних продуктів.

Для досягнення поставленої мети проекту були визначені наступні задачі:

- провести детальний аналіз предметної області та існуючих рішень реалізації, визначити переваги та недоліки для кожного рішення для формування мети та задач проекту;
- проаналізувати та вибрати інструменти та методи дослідження;
- виконати планування IT-проекту, створивши план роботи проекту с прив’язкою до часу (діаграму Ганта) та оцінки ризиків;
- виконати збір даних про помилки та виправлення використання бібліотек машинного навчання, нормалізація даних;
- визначити характеристики помилок та виправлень використання бібліотек машинного навчання, визначення патернів та антипатернів;
- створення моделі та інформаційної технології для аналізу використання бібліотек машинного навчання;
- реалізація моделі та інформаційної технології, тестування та перевірка точності моделі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд використання бібліотек машинного навчання в життєвому циклі розробки програмного забезпечення

Машинне навчання проникає у всі пов'язані з ними бізнес-процеси, щоб вивести їх на якісно новий рівень. Майже будь-який постачальник послуг з розробки програмного забезпечення шукає шляхи прискорення життєвого циклу розробки програмного забезпечення за допомогою використання машинного навчання. Можливості машинного навчання можуть бути додані до системи кількома способами, включаючи програмні системи з компонентами та структурами, інструменти та бібліотеки, що забезпечують функціональність використання. Більш детальний огляд використання машинного навчання в життєвому циклі розробки програмних продуктів представлений у даному розділі.

Вимоги проекту

Управління вимогами - процес збору, перевірки та відстеження того, що необхідно кінцевим користувачам від частини програмного забезпечення - є основною причиною затримок, дорогих або невдалих проектів, коли вони виконуються погано [12]. Деякі вендори впровадили цифрових помічників, які можуть аналізувати документи з вимогами, неясності і невідповідності прапорів і пропонувати поліпшення. Ці інструменти можуть виявляти неточності або інші недоліки - такі, як неповні вимоги, невимовний кількісний аналіз (відсутні одиниці або допуски), складові вимоги і аварійні застереження, - з тим щоб прискорити розгляд вимог. Згідно з повідомленнями, підприємства, що використовують такі інструменти, змогли скоротити час розгляду вимог більш ніж на 50 відсотків.

Виявлення проблем

Ідентифікація проблеми є основним процесом в кожному бізнесі і робить подальші кроки щодо її вирішення. В кінцевому рахунку, потреба в розробці програмного забезпечення починається вже на цьому етапі.

Сильні і слабкі сторони бізнесу повинні бути проаналізовані належним чином. Поряд з цим враховується зворотний зв'язок від клієнтів, співробітників, зацікавлених сторін і т.д.

Наука про дані може допомогти особам, які приймають рішення, двома способами: за допомогою інструментів бізнес-аналітики або за допомогою машинного навчання. Доступ до бізнес-даних можна отримати з панелі управління бізнес-аналітики, де різні бізнес-показники відображаються у вигляді карт зі значенням продуктивності.

Карти з низькою продуктивністю викликають проблеми в бізнесі. Досить простий спосіб визначення відносних причин. Або ж машинне навчання, де машина отримує всі бізнес-дані в структурованому або не структурованому вигляді. Потім машина аналізує алгоритм і забезпечує розуміння. Цих знань досить для ідентифікації проблем.

Документація

Коли ідея сформульована, наступним важливим етапом є складання документації на основі плану. Планування - це докладний звіт, який повинен відображати кінцеву мету вимог.

Тут слід згадати про вартість, розподіл ресурсів, терміни і про все. Цей план можна розглядати як запобіжний документ, в якому кожну вимогу пропонується зацікавленим сторонам. Автоматична класифікація документів на основі контрольних показників, таких як витрати, ресурси і т.д., може бути досягнута за допомогою машинного навчання.

Після подачі на машину даних з першого етапу, бажана мета встановлюється в якості даних, що вводяться. Потім система розраховує інше і прогнозує значення для кожного контрольного показника.

Для розподілу ресурсів враховується аналіз поведінки співробітників, а потім розподіляється відповідно до їх здібностей. Графік встановлюється на основі навичок кожного ресурсу і їх минулих здібностей завершення мети з організацією.

Прототип програмного забезпечення

Після затвердження плану відповідної командою створюється прототип або концепція програмного забезпечення. Він проектується відповідним чином з використанням шаблонів архітектури та розробки програмного забезпечення.

Концептуальна модель розглядається як перший крок в розробці програмного забезпечення. Архітектура програмного забезпечення будується на основі потоку програмного забезпечення, підготовленого на етапі документування.

Машини можуть перевірити висновок моделей концепції, вивчивши наперед задані еталони. Для кожного пункту можна задати значення, наприклад, надійність вибору мови кодування, недоліки і багато іншого.

Кодування

На цьому етапі розробляється структура кодування з усіма функціями відповідно до вимог. Після прийняття концептуальної моделі йде гнучкий процес. Максимізація результату шляхом управління графіком вважається дуже важливою в цей період.

Правильні мови програмування використовуються для забезпечення стабільної системи з меншою кількістю помилок або їх відсутністю. Однак помилки трапляються, і найкраща практика, якої слід дотримуватися, полягає в тому, щоб зробити їх якомога менше.

Наука про дані полегшує зусилля розробників, надаючи платформу програмування, яка володіє машинним навчанням. Ці платформи надають автоматично згенерували пропозиції по коду при підготовці коду. Надлишок добре підтримується в кожній задачі розробки. Сприяння такому сприянню надають Kite [13], Codota [14] і ін. Слідуючи цьому методу, компанії можуть

зробити своїх розробників активними, а також підвищити продуктивність праці.

Тестування

Тестування коду - наступне завдання після етапу розробки, де перевіряється і оптимізується вся структура кодування. На цьому етапі, крім тестування коду, розглядаються і деякі інші аспекти.

Тут перевіряється і порівнюється з таблицею вимог або документацією забезпечення якості програмного забезпечення, потік кодування, масштабованість для майбутніх налаштувань, дефекти і т.д.

Тестери програмного забезпечення, як вручну, так і за допомогою програмного забезпечення, забезпечують досягнення цілей програмного забезпечення. Розробники повинні виправити всі помилки, про які повідомляє команда тестувальників, і внести зміни для забезпечення високої продуктивності.

В якості даних в цьому процесі розглядається історичний аналіз і попередні проекти, засновані на різних вимогах замовника. Команда тестувальників і розробники отримують можливість подавати помилки або баги на машину.

Розгортання

Після діагностики система, нарешті, досягає стану рівноваги, тобто стабільної програмної платформи. Життєвий цикл розробки програмного забезпечення відповідає остаточного станом випуску програмного забезпечення після успішного завершення попередніх етапів.

Помилки, які все ще з'являються в програмному забезпеченні, виправляються якомога швидше. На цьому етапі життєвого циклу розробки програмного забезпечення збирається і впроваджується зворотний зв'язок із замовниками.

Підключивши машинне навчання, то цілком зрозуміло, що статус впровадження також може бути оброблений за допомогою машин. Збираються

і аналізуються нові відгуки замовників для створення нових пропозицій для розробників.

Функціональність і обслуговування

Супровід програмного забезпечення також відомо як "стадія підтримки" циклу розробки програмного забезпечення.

Якщо система досягає більш високого рівня застосовності, коли більше не потрібно оновлення, то, як кажуть, вона досягла стадії "насичення".

Машинам потрібні дані, сучасні технології, ринкові виразки і злети, акції, в основному, все, що можна отримати через Інтернет.

В даний час навіть неструктуровані форми даних аналізуються за допомогою Natural Language Processing (NLP) [15]. Аналізуючи існуючі дані з новими вимогами, машина запропонує нові інновації, які можуть бути інтегровані з програмним забезпеченням.

1.2 Огляд проблем використання бібліотек машинного навчання

Розробники часто стикаються з різними типами помилок при спробі використання машинного навчання в реалізації програмного продукту. Щоб зрозуміти помилки використання бібліотек машинного навчання, їх було проаналізовано та розділено на різні категорії. Класифікація заснована на [14] і адаптована на основі всіх проаналізованих постів з Stack Overflow. В кожному записі можна було виділити тільки одну помилку. Якщо в пості розглядається декілька категорій помилок, то береться за увагу та помилка, яка має більший пріоритет в питанні.

Помилка API

Ця група помилок викликана самим API бібліотеки. Зазвичай, коли розробник використовує API бібліотеки, різні помилки, пов'язані з цим API, успадковуються автоматично без відома користувача. Основні причини появи

помилок API можуть бути через зміни визначення API в різних версіях, відсутність сумісності між API і іноді неправильної або заплутаною документації.

Помилка кодування

Подібні помилки виникають через помилки програмування. Це, в свою чергу, призводить до появи інших типів помилок в програмному забезпеченні, які призводять або до помилки виконання, або до невірних результатів. Великий відсоток перевірених помилок виникає через синтаксичні помилки, які не можна виправити, змінивши лише деякі рядки коду. Цей тип помилок не виявляється компілятором мови програмування, що призводить до неправильного висновку.

Помилка даних

Ця помилка може виникнути, якщо вхідні дані неправильно відформатовані або неправильно очищені перед передачею їх в модель глибокого навчання. Цей тип помилки виникає до того, як дані надходять в модель глибокого навчання. Це відбувається не через неправильну моделі машинного навчання, а виключно через типи і структури навчальних або тестових даних. Про помилки даних зазвичай позначаються компілятором, але в деяких сценаріях вони можуть проходити без перевірки в процесі компіляції і генерувати помилкові результати.

Структурна помилка (SB)

Переважна більшість помилок машинного навчання виникає через неправильні визначень структури моделі. До них відносяться розбіжність розмірів між різними рівнями моделей машинного навчання, наявність аномалії між навчальними і тестовими наборами даних, використання неправильних структур даних при реалізації конкретної функції і т. Д. Помилки цього типу можна розділити на чотири підкатегорії.

Помилка управління і послідовності

Цей підклас помилки викликаний неправильною структурою потоку управління. У багатьох сценаріях через неправильне умови if-else модель не

працює належним чином. Цей тип помилок або призводить до збою, коли частина моделі глибокого навчання не працює, або призводить до неправильної роботи через неправильну обробки даних через рівні.

Помилка потоку даних

Основна відмінність між помилкою потоку даних і помилкою даних полягає в місці походження. Якщо помилка виникає через невідповідність типу або форми вхідних даних після того, як вони були передані в модель глибокого навчання – то це помилка потоку даних. Вони включають ті сценарії, в яких шари моделі не узгоджені з-за різної форми даних, що використовуються в послідовних шарах. Щоб виправити ці помилки, розробникам необхідно змінити модель або змінити форму даних.

Помилка ініціалізації

У глибокому навчанні помилка ініціалізації означає, що параметри або функції не ініціалізується належним чином перед використанням. Цей тип помилок не обов'язково призведе до помилки виконання, але просто погіршить роботу моделі. Тут визначення функцій включає як визначені користувачем, так і певні API. Ми також відносимо помилку до цієї категорії, якщо API НЕ ініціалізується належним чином.

Логічна помилка

У глибокому навчанні логічне розуміння кожного етапу конвеєра є невід'ємною частиною процесу кодування. При неправильній логічній структурі моделі глибокого навчання вихідні дані програми можуть привести або до помилки виконання, або до помилкового результату. Ці помилки часто виникають при відсутності в коді належних умов захисту.

Помилка обробки

Одним з найбільш важливих рішень в структурі моделі глибокого навчання є вибір правильного алгоритму процесу навчання. Фактично, різні алгоритми глибокого навчання можуть призводити до різної продуктивності і різних результатів [30]. Крім того, щоб різні рівні були сумісні один з одним,

типи даних кожного рівня повинні відповідати контрактами між ними. Помилки обробки відбуваються через порушення цих контрактів.

Не модельна структура помилка (NMSB)

На відміну від SB, NMSB виникають поза стадії моделювання. Іншими словами, ця помилка може статися в будь-якому глибокому етапу навчання, такого як етап навчання або етап прогнозування, за винятком етапу моделювання. **NMSB** має аналогічні підкатегорії як SB.

Помилка управління і послідовності

Цей підклас аналогічний помилці управління і послідовності в SB. Помилка викликана неправильною структурою потоку управління, наприклад неправильним умовою if-else ; проте ця помилка відбувається поза стадії моделювання.

Помилка ініціалізації

Цей підклас аналогічний помилку ініціалізації в SB. Помилка викликана не правільною ініціалізацією параметра або функції перед її використанням.

Логічна помилка

Цей підклас схожий на логічну помилку в SB. Помилка викликана неправильним розумінням поведінки операторів case і логічних операторів.

Помилка обробки

Цей підклас схожий на помилку обробки в SB. Помилка викликана не правильно обраним алгоритмом.

1.3 Дослідження існуючих бібліотек машинного навчання

В роботі розглянуто і проаналізовано найбільш вживані на сьогодні бібліотеки машинного навчання, а саме pandas, Matplotlib, Tableau, NumPy, scikit-learn, NLTK, TensorFlow, TensorBoard, PyTorch, Keras, Caffe, Apache Spark.

Pandas

Бібліотека розроблена, щоб зробити аналіз і моделювання даних зручним на Python. pandas спрощує аналіз, перетворюючи файли даних CSV, JSON і TSV або базу даних SQL в фрейм даних, об'єкт Python, схожий на Excel або SPSS-таблицю з рядками і стовпцями. Більш того, pandas комбінуються з інструментарієм IPython і іншими бібліотеками для підвищення продуктивності і підтримки спільної роботи.

Matplotlib

Matplotlib - це 2D бібліотека Python для креслення. Незважаючи на те, що бібліотека написана в основному на Python, вона розширена за допомогою NumPy і іншого коду, тому працює добре навіть при використанні для великих масивів.

Matplotlib дозволяє генерувати візуалізації виробничого якості з кількома рядками коду. Функціональність бібліотеки може бути розширена сторонніми пакетами візуалізації, такими як seaborn, ggplot і HoloViews. Фахівці також можуть додавати додаткові функції, використовуючи набори інструментів для проекції і картографії Basemap і cartopy.

Tableau

Tableau - це інструмент візуалізації даних, що використовується в науці про дані і бізнес-аналітиці [16]. Ряд специфічних особливостей роблять це програмне забезпечення ефективним для вирішення завдань в різних галузях промисловості і інформаційних середовищах.

Досліджуючи і відкриваючи дані, програмне забезпечення Tableau швидко витягує з них інформацію і представляє її у зрозумілих форматах. Воно не вимагає відмінних навичок програмування і може бути легко встановлено на всі види пристроїв. У той час як невеликий сценарій повинен бути написаний, більшість операцій виконується перетягуванням.

NumPy

Раніше згадуваний NumPy є пакетом розширень для виконання чисельних обчислень за допомогою Python, який замінив NumArray і Numeric [17]. Він підтримує багатовимірні масиви (таблиці) і матриці. Дані машинного навчання представлені в масивах. А матриця - це двовимірний масив чисел. NumPy містить широкомовні функції як інструменти для інтеграції Ci / Ci ++ і коду Fortran. Її функціональність також включає в себе перетворення Фур'є, лінійну алгебру і можливості роботи з випадковими числами.

Фахівці в області даних можуть використовувати NumPy в якості ефективного контейнера для зберігання багатовимірних загальних даних. Завдяки можливості визначення довільних типів даних, NumPy легко і швидко інтегрується з численними типами баз даних.

scikit-learn

scikit-learn - це бібліотека з відкритим вихідним кодом для машинного навчання на Python, побудована на основі SciPy (Scientific Python), NumPy і matplotlib.

Бібліотека призначена для виробничого використання. Простота, низькоякісний продукт, можливості спільної роботи, продуктивність і велика документація, написана простою мовою, сприяють її популярності серед різних фахівців.

Scikit-learn надає користувачам ряд усталених алгоритмів для контрольованого і неконтрольованого навчання. Бібліотека фокусується на моделюванні даних, а не на їх завантаженні, маніпулюванні і узагальненні.

NLTK

NLTK - це практично стандартна бібліотека на Python для роботи з текстом, що володіє безліччю корисних функцій[18]. Наприклад, різні типи тексту, обробка пропозицій і слів, частина мовних тегів, аналіз структури пропозиції, розпізнавання іменованих об'єктів, класифікація тексту, аналіз почуттів і багато іншого.

TensorFlow

Важливою особливістю цієї бібліотеки є те, що чисельні обчислення виконуються за допомогою графіків потоку даних, що складається з вузлів і ребер [19]. Вузли представляють собою математичні операції, а ребра - це багатовимірні масиви даних або тензори, над якими виконуються ці операції.

TensorFlow відрізняється гнучкістю і може використовуватися на різних обчислювальних платформах (CPU, GPU і TPU) і пристроях, від настільних комп'ютерів до кластерів серверів і мобільних і прикордонних систем. Ще однією перевагою цього фреймворка є те, що він працює як для досліджень, так і для повторюваних завдань машинного навчання.

TensorBoard

TensorBoard - це набір інструментів для графічного представлення різних аспектів і етапів машинного навчання в TensorFlow [20].

TensorBoard читає файли подій TensorFlow, що містять зведені дані (спостереження за конкретними операціями моделі), що генеруються під час роботи TensorFlow.

Структура моделі, показана на графіках, дозволяє дослідникам переконатися в тому, що компоненти моделі розташовані там, де це необхідно, і правильно підключені.

За допомогою графічного візуалізатора користувачі можуть досліджувати різні верстви абстракції моделі, збільшуючи і зменшуючи масштаб будь-якій частині схеми. Ще однією важливою перевагою візуалізації TensorBoard є те, що вузли одних і тих же типів і схожих структур фарбуються одними і тими ж квітами. Користувачі також можуть переглядати розмальовку за приладами (CPU, GPU або поєднання обох), виділяти конкретний вузол з функцією "трасування входів" і візуалізувати одну або кілька схем одночасно.

TensorBoard показує метрики під час розробки моделі і дозволяє прийняти рішення щодо моделі. Наприклад, дуже зручно спостерігати за тим, як працює модель, при підстроювання її гіперпараметрів і виборі тієї, яка працює найкраще.

Крім відображення метрик продуктивності, TensorBoard може показувати користувачам безліч іншої інформації, такої як гістограми, аудіо, текстові та графічні дані, розподілу, вбудовування та скаляри.

PyTorch

PyTorch - це система машинного навчання для глибоких нейронних мереж, яка підтримує і прискорює роботу GPU.

PyTorch був розроблений з ідеєю забезпечення максимально швидкого і гнучкого моделювання. Варто зазначити, що робочий процес в PyTorch схожий на процес в NumPy, наукової обчислювальної бібліотеці, заснованої на Python.

Динамічний обчислювальний граф - одна з особливостей, які роблять цю бібліотеку популярної. У більшості фреймворків, таких як TensorFlow, Theano, CNTK і Caffe, моделі побудовані статичним способом. Вчений, що займається даними, повинен змінити всю структуру нейронної мережі - перебудувати її з нуля - щоб змінити її поведінку. PyTorch робить це легше і швидше. Фреймворк дозволяє довільно змінювати поведінку мережі без затримок і накладних витрат.

Keras

Keras - дає можливість вченим, які працюють з даними, швидко проводити експерименти по машинному навчання.

Бібліотека може працювати на GPU і CPU і підтримує як повторювані мережі, а також їх комбінації. Швидке прототипування можливо за допомогою високорівневого, зрозумілого інтерфейсу бібліотеки, розбиття мереж на послідовності окремих модулів, які легко створювати і додавати. На думку фахівців з даними, швидкість моделювання є однією з сильних сторін даної бібліотеки.

Apache Spark

Apache Spark - це розподілене фреймворк для кластерних обчислень, який зазвичай оснащений двигуном обробки даних в пам'яті. Функціональність цього «двигуна» включає в себе ETL (витяг, перетворення і

завантаження), машинне навчання, аналіз даних, пакетну обробку і потокову обробку даних.

Цей інструмент використовує концепцію потокової обробки даних для розподілених обчислень і дозволяє масштабувати рішення для великих кластерів.

Різноманітність варіантів запуску (локально, в кластерах, хмарі або на місці) і можливість доступу до даних з будь-яких джерел - це ще одна корисна особливість Apache Spark.

Caffe

Caffe2 - для спрощення і гнучкого глибокого вивчення складних моделей і підтримки мобільного розгортання.

У користувачів є кілька варіантів організації обчислень за допомогою бібліотеки, яка може бути встановлена і запущена на робочому столі, в хмарі або в центрі обробки даних.

Розгорнуті моделі можуть швидко працювати на мобільних пристроях завдяки інтеграції з IDE Xcode, Visual Studio і Android Studio. Цей фреймворк також дозволяє швидко збільшувати або зменшувати масштаб, не вдаючись до рефакторингу дизайну.

Швидке прототипування, дослідження і розробки є перевагами використання Caffe.

У реалізації програмного продукту використовують цю бібліотеку, тому що він має чітку інфраструктуру коду, і його легко розширити для дослідження нових методів.

1.4 Порівняння алгоритмів обчислення подібності тексту

Інформація про результати використання бібліотек машинного навчання у реалізації проектів програмних продуктів буде братися з текстових повідомлені на Stack Overflow, тому потрібно розглянути підходи до обробки

та аналізу текстових повідомлень. Для даної роботи було вибрано алгоритм порівняння текстів, як головних визначник результату.

Найважливішим етапом в роботі питально відповідної системи є обробка тексту. Але щоб зрозуміти, як саме формуються відповіді на питання, необхідно розглянути основи Natural Language Processing (NLP). У розробці питально-відповідних систем розділ Natural Language Processing присвячений тому, як комп'ютери аналізують природні мови і дозволяє застосовувати алгоритми машинного навчання для тексту й мови. NLP для тексту включає в себе кілька процесів. Кожен з них по-своєму важливий і необхідний при створенні запитань-відповідей системи [21].

Поділ тексту за пропозиціями. Ідея полягає в тому, що в англійській і деяких інших мовах, можна виділяти нову пропозицію кожен раз, коли знаходиться певний знак пунктуації - крапка.

Лематизація. Процес приведення будь-якої форми слова до його словникової формі. Слово drove має словникову форму drive.

Стемінг. Це процес, в якому відрізається все «зайве» від кореня слова. Наприклад, слова cat, cats, cat's зводяться до однієї форми - cat.

Стоп-слово. Такі слова виключаються з тексту до або після обробки. Зазвичай це артиклі, союзи і т.д., так як вони не несуть ніякого смислового навантаження, а лише створюють шум в процесі обробки.

Регулярні вирази. Деяка послідовність символів, яка визначає шаблон пошуку. Іншими словами, для пошуку використовується рядок-зразок, що складається з символів і мета-символів, яка задає правило пошуку.

Мішок слів. Це техніка, при використанні якої на природній мові кожен документ або текст виглядає як невпорядкований набір слів без будь-якої інформації про зв'язок між ними. Будь-яка інформація про порядок або структуру слів ігнорується, звідси і назва.

Кожен з цих процесів допомагає зрозуміти, як краще розробити питально-відповідну систему [22]. Адже знання основи виділення ознак

дозволить використовувати їх як вхідні дані для алгоритмів машинного навчання.

Алгоритми розпізнавання схожості тексту

Для розробки питально-відповідних систем використовуються різні алгоритми, які здатні розпізнавати на скільки тексти схожі один на одного. Вони дозволяють оцінити, чи правильно система відповідає на поставлені їй питання. Найбільш відомими алгоритмами є коефіцієнт Жаккар і коефіцієнт Серенсена.

Коефіцієнт Жаккар.

Інакше називається мірою Жаккар або коефіцієнтом подібності. Заснований на використанні інформації про безліч загальних символів [23]. Обчислюється, як відношення кількості унікальних символів в двох множинах до загальної кількості унікальних символів в двох множинах (формула 1.1).

$$J(AB) = \frac{|A \cap B|}{|A \cup B|} \tag{1.1}$$

Для найкращого розуміння роботи цього алгоритму, слід розглянути приклад. Одна людина робить запит і пише в пошуковому рядку слово «молоко». Друга людина виявляється менш грамотною і припускається помилки, написавши в пошуковому рядку «малок». За змістом запити абсолютно однакові, адже обидва мали на увазі один і той же напій, але написання виявилося різним. Питально-відповідна система, у свою чергу, повинна якимось чином інтерпретувати орфографічно неправильний запит. Щоб це зробити, їй необхідно оцінити, виміряти схожість запитів, видавши кількісну оцінку ступеня цієї схожості. Саме тут на допомогу приходить коефіцієнт Жаккар.

Індекс Соренсена

Бінарна міра подібності, еквівалентна (пов'язана однієї монотонно зростаючою залежністю) міру Жаккар [24]. По суті, вони обидва еквівалентні в тому сенсі, що, з огляду на значення коефіцієнта Серенсена S, можна розрахувати відповідне значення індексу Жакар J і навпаки, використовуючи рівняння (формула 1.2).

$$J = S / (2-S) \text{ і } S = 2J / (1 + J) \quad (1.2)$$

Алгоритм Ретклiфф-Обершельпа

Подібність двох рядків S_1 і S_2 визначається за формулою: подвійну кількість співпадаючих символів K_m ділиться на загальну кількість символів в обох текстах [25]. Відбувається пошук максимальної суміжнії послідовності двох рядків, після чого алгоритм запускається рекурсивно для решти рядків, розташованих ліворуч і праворуч від знайденої (формула 1.3).

$$D_{ro} = \frac{2K_m}{|S_1|+|S_2|} \quad (1.3)$$

Дані алгоритми будуть порівнюватися. Щоб побачити, як працюють алгоритми, була взята невелика базу даних. У ній містяться питання і відповіді на них, а також перефразовані питання, які за змістом ідентичні вихідним, але складаються з інших слів. Всі символи тексту наводяться до нижнього регістра, видаляються знаки пунктуації, різні небуквених символи і цифри. Кожен рядок розбивається на список складових її слів і видаляються стоп-слова, а також використовується лематизації.

У програмі порівнюються два питання, схожих між собою і до них застосовуються вищевказані алгоритми. В кінці виводиться відсоток порівняння. Чим він вищий, тим краще працює той чи інший алгоритм. Результатом програми є графік порівняння роботи двох алгоритмів з

алгоритмом Ретклiффа-Обершельпа (рис. 1.1), виходячи з якого видно, що алгоритм Серенсена і Жаккар не сильно відрізняються один від одного, але явно поступаються алгоритму Ретклiффа-Обершельпа. Тому для порівняння в двох текстів був вибраний саме алгоритм Ретклiффа-Обершельпа.

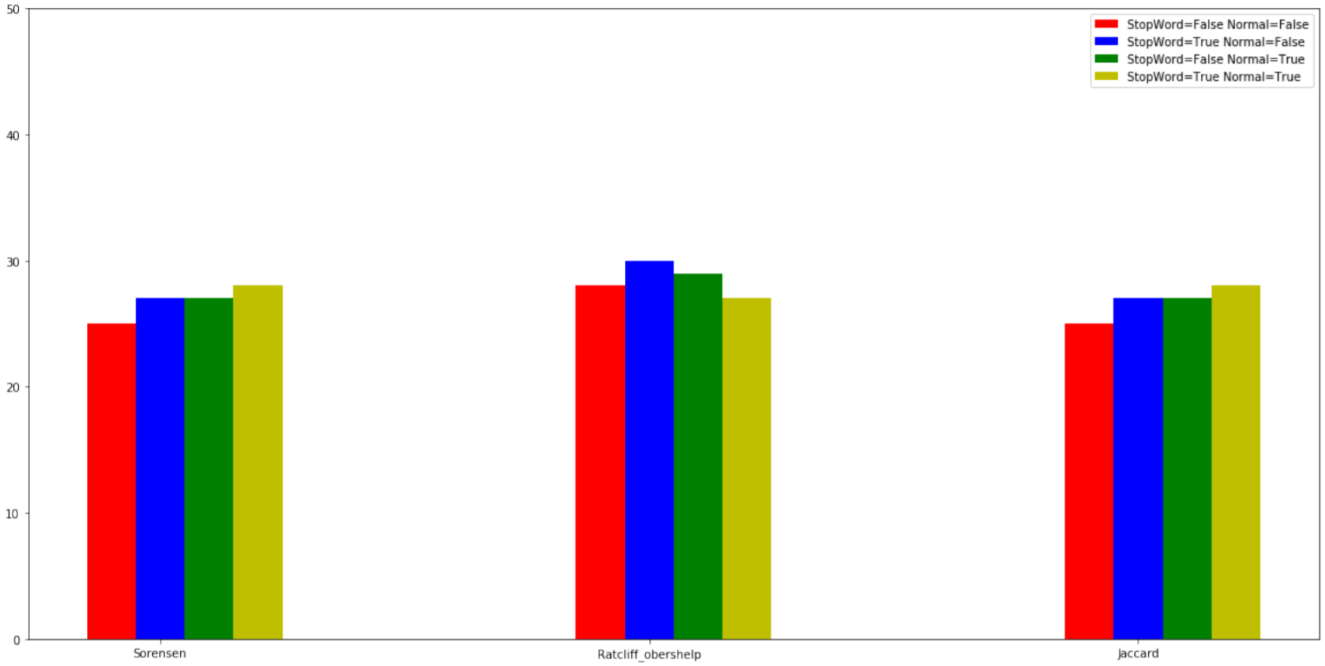

Рисунок 1.1 – Результат порівняння алгоритмів

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Ціллю даного проекту є створення моделі та інформаційної технології аналізу результатів використання бібліотек машинного навчання у реалізації розробки програмних продуктів. Модель та інформаційна технологія має аналізувати використання бібліотек машинного навчання в реалізації розробки програмних продуктів та повідомляти про зловживання бібліотекою, виявлення помилок реалізації, які попередньо були вирішені.

Мета створення моделі та інформаційної технології аналізу – надати розробникам можливість виявити помилки використання бібліотек машинного навчання на стадії розробки та виявити зловживання використання бібліотек.

Надалі модель може бути інтегрована в системи програмних засобів, для швидкого та зручного виявлення помилок використання бібліотек.

Модель та інформаційна технологія повинна виконувати перелік наступних задач:

- аналізувати помилки та виправлення популярної системи та відповідей для професійних програмістів Stack Overflow;
- розпізнавати код розробників;
- виявляти раніше відомої помилки в коді;
- пропонувати виправлення помилки допущеної розробником.

2.2 Засоби реалізації проекту

Мова програмування Python - високорівнева універсальна інтерпретована мова сценаріїв. При розробці мовою Python велика увага приділяється простоті і зрозумілості синтаксису, що не тільки скорочує час вивчення його основ, але і підвищує швидкість розробки в цілому [26]. Це далеко не всі переваги даного мови, основні з них:

- об'єктно-орієнтованість;
- вільне поширення і широка підтримка;
- кросплатформеність;
- розвинені функціональні можливості.

Мова Python об'єднує дві парадигми програмування - об'єктно, яка є потужним засобом структурованого програмного коду для багаторазового використання, і процедурну, що розширює коло вирішення завдань, дозволяючи використовувати засоби Python при вирішенні тактичних завдань з відсутністю фази проектування. Об'єктна модель Python підтримує поняття поліморфізм, перевантаження операторів і множинне спадкування.

Кросплатформенність мови досягається завдяки його реалізації на переносимій ANSI C, що дозволяє програмам, написаним на мові Python, однаково добре компілюватиметься і виконуватися на будь-яких платформах, де встановлена сумісна версія Python.

Гібридна природа Python об'єднує в собі простоту і зручність мов сценаріїв і потужності компілює мов, що робить Python зручним засобом розробки додатків різного типу. Однак найбільша ефективність мови досягається при вирішенні задач аналізу даних і автоматизації процесів. Python широко використовується в дослідницьких роботах. Мова програмування Python має потужний вбудованим інструментарієм (вбудовані типи об'єктів і динамічна типізація, автоматичне керування пам'яттю) і

можливістю використання зовнішніх бібліотек і утиліт сторонніх розробників для вирішення більш вузькоспеціалізованих завдань.

Jupyter Notebook - це веб-додаток з відкритим вихідним кодом, що дозволяє створювати коди і документи і обмінюватися ними [27].

Воно надає середовище, в якому можна документувати код, запускати його, переглядати результати, візуалізувати дані і бачити результати, не виходячи з середовища. Це робить його зручним інструментом для виконання від кінця до кінця робочих процесів в області науки про дані - очищення даних, статистичного моделювання, побудови і навчання машинним моделям, візуалізації даних і багатьох, багатьох інших застосувань.

Ефективний інструмент навчання: ноутбук Jupyter - це не тільки інструмент для наукових досліджень і аналізу даних, а й відмінний засіб навчання. Прикладом може служити IPython Blocks - бібліотека, що дозволяє вам або вашим учням створювати сітки з різнокольорових блоків.

Інтерактивний код і дослідження даних: пакет ipywidgets надає безліч спільних елементів управління призначеним для користувача інтерфейсом для інтерактивного вивчення коду і даних.

3 ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АНАЛІЗУ РЕЗУЛЬТАТІВ ВИКОРИСТАННЯ БІБЛІОТЕК МАШИННОГО НАВЧАННЯ У РЕАЛІЗАЦІЇ ПРОЕКТІВ РОЗРОБКИ ПРОГРАМНИХ ПРОДУКТІВ

3.1 Концептуальна модель інформаційної технології аналізу результатів використання бібліотек машинного навчання

Концептуальна модель представляє собою ядро програми дослідження. У ряді випадків написання програми може починатися зі спроби побудувати концептуальну модель і потім, в процесі роботи над моделлю, сформулювати мету і завдання, визначити об'єкт та предмет дослідження [28].

Послідовність роботи в даному випадку буде здаватися характером дослідницької проблеми та наявним обсягом знань з даної проблеми, наявністю теорій, які можуть бути застосовані для опису проблеми та індивідуальним стилем дослідницької роботи.

Концептуальна модель - це сукупність взаємопов'язаних понять, що лежать в основі дослідницького дизайну, системний опис досліджуваної області. Це може бути простий список понять і їх можливих зв'язків або більш розроблена схематична діаграма ключових впливів, передбачуваних взаємозв'язків, можливих рішень дослідницької проблеми. Концептуальна модель наочно описує структуру модельованої предметної області і зв'язки між її елементами.

Концептуальна модель представляє в графічній або оповідної формі основні моменти, що вимагають вивчення, - чинники, конструкти, змінні і їх ймовірні зв'язки один з одним. Вона структурує процес дослідження, дозволяє пов'язати теорію і емпіричні дані, направляє збір даних і їх подальшу інтерпретацію.

Концептуальна модель допомагає вирішити, які змінні найбільш важливі, які з цим найбільш значимі і, отже, яку інформацію необхідно збирати. Концептуальна модель, позначаючи теоретичний контекст, в рамки якого може бути поміщена досліджувана ситуація, дає можливість чіткої постановки дослідницьких питань і розробки гіпотез дослідження.

Для даної роботи була створена концептуальна модель інформаційної технології аналізу результатів використання бібліотек машинного навчання, яка представлена на рис. 3.1.

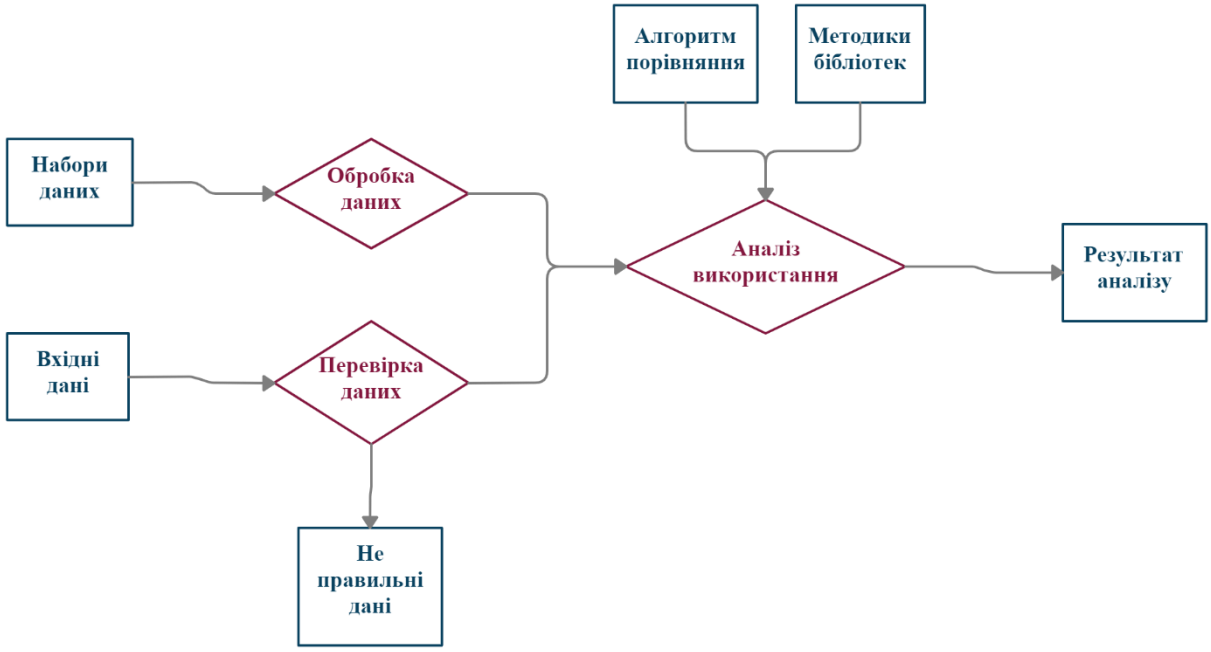

Рисунок 3.1 – Концептуальна модель інформаційної технології аналізу результатів використання бібліотек машинного навчання

3.2 Структурно-функціональне моделювання

IDEF 0 реалізує методику функціонального моделювання складних систем. Найбільш відомою реалізацією IDEF0 є методологія SADT (Structured Analysis and Design Technique). Функціональна модель IDEF0 являє собою набір блоків, кожен з яких представляє собою «чорний ящик» з входами і виходами, управлінням та механізмами, які деталізуються (декомпозуються) до необхідного рівня. Найбільш важлива функція розташована у верхньому лівому кутку. З'єднуються функції між собою за допомогою стрілок і описів функціональних блоків [29]. При цьому кожен вид стрілки або активності має власне значення. Дана модель дозволяє описати всі основні види процесів, як адміністративні, так і організаційні. Стрілки можуть бути: - Вхідні - вступні, які ставлять певне завдання. - Вихідні - виводять результат діяльності. - Керуючі (зверху вниз) - механізми управління (положення, інструкції тощо). - Механізми (від низу до верху) - що використовується для того, щоб зробити необхідну роботу. Діаграми декомпозиції призначені для деталізації функцій і виходять при розбиванні контекстної діаграми на великі підсистеми (функціональна декомпозиція) і описують кожну підсистему та їх взаємодію. Єдина функція, представлена на діаграмі контекстного верхнього рівня, може бути розкладена на основні під функції за допомогою створення дочірньої діаграми.

Структурно-функціональна модель першого рівня має наступний вигляд, (рис.3.2 – рис.3.4). Декомпозиція блоків «Робота з даними» та «Створення моделі порівняння» представлені на рисунках Б.1 – Б.2. Цілю даної структурно-функціональної моделі є показати роботу моделі аналізу результатів використання бібліотек машинного навчання у реалізації проектів розробки програмних продуктів. Вхідними даними є «Набір даних» та «Дані

роботи». «Набір даних» - містить питання та відповіді із Stack Overflow на яких буде навчатися та тестуватися модель. «Дані роботи» - це дані використання бібліотек машинного навчання. На виході маємо результат використання даних роботи.

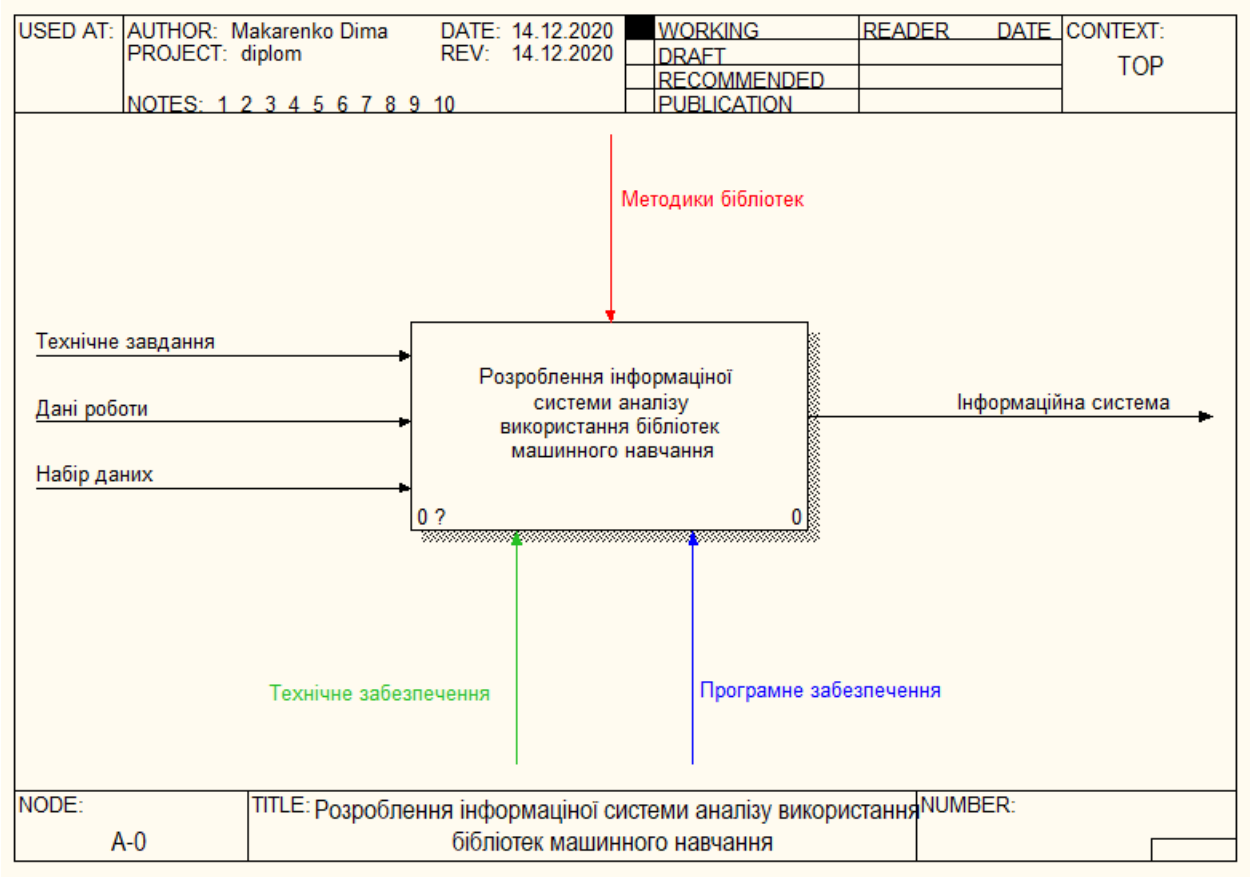

Рисунок 3.2 – Структурно-функціональна модель

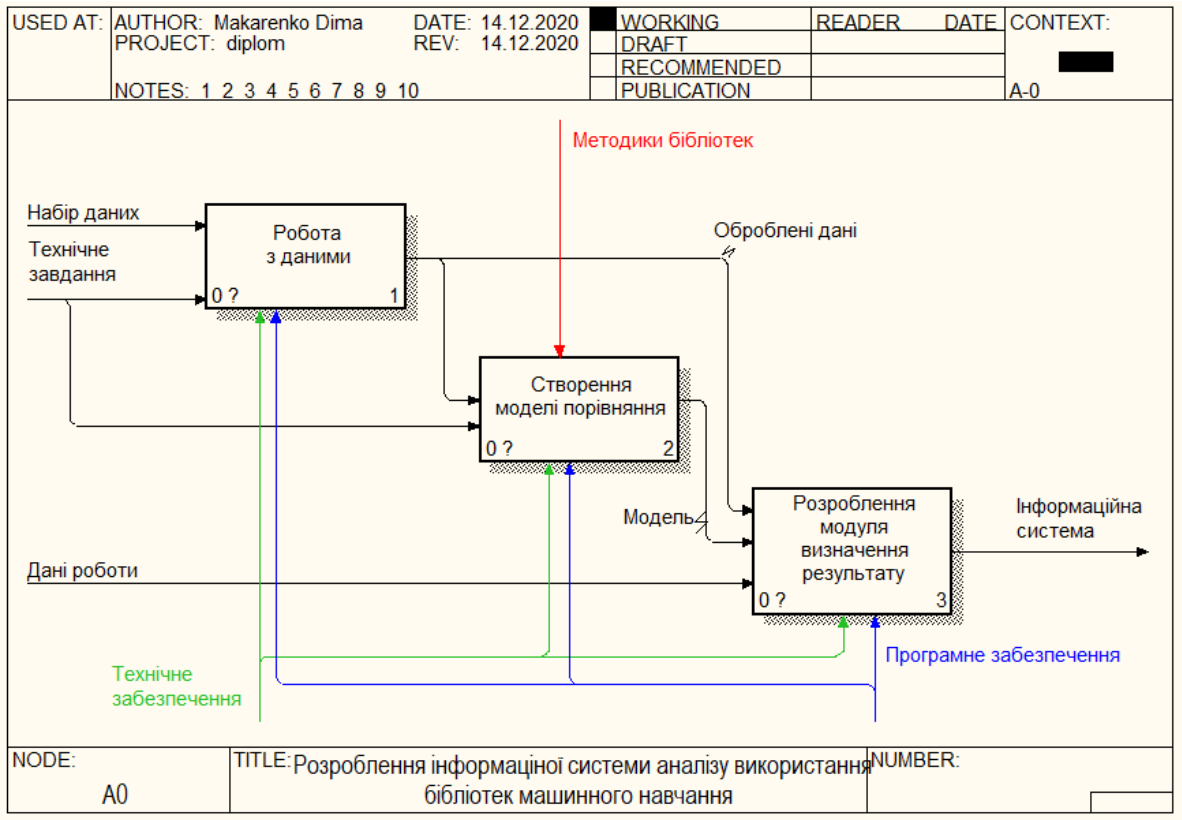

Рисунок 3.3 – Перший рівень декомпозиції

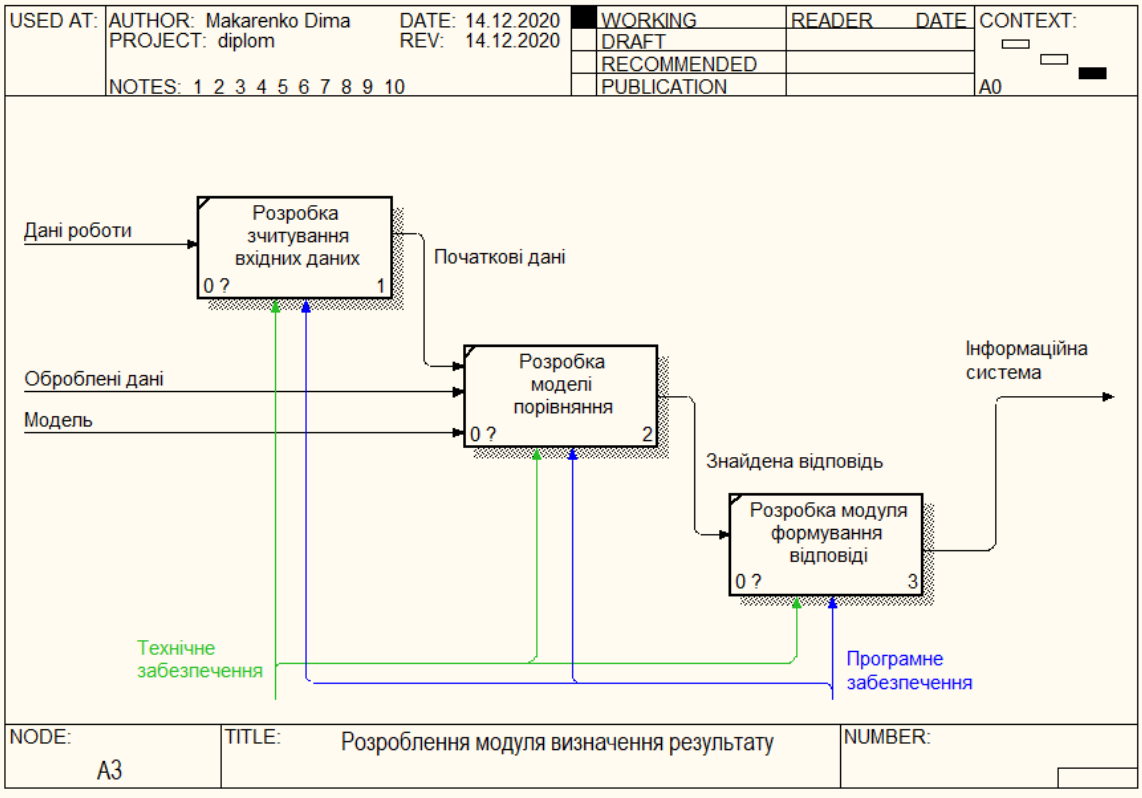

Рисунок 3.4 – Декомпозиція блоку «Визначення результату використання»

4 ШАБЛОНИ ПОМИЛОК ТА ВИПРАВЛЕНЬ ВИКОРИСТАННЯ БІБЛІОТЕК МАШИННОГО НАВЧАННЯ У РЕАЛІЗАЦІЇ ПРОЕКТІВ РОЗРОБКИ ПРОГРАМНИХ ПРОДУКТІВ

У більшості випадках розробники, які використовують машинне навчання, не розуміють з якою саме проблемою вони зіткнулися. Це уповільнює розробку, саме тому потрібні дослідження в даній області. Вони допоможуть дізнатися основні причини помилок, слабкі місця бібліотек та інше. Такі дослідження можна використовувати для навчання використання бібліотеками, або для подальшого їх покращення.

Цей розділ представляє комплексе дослідження проблем, помилок та виправлень, з якими стикаються розробник. Дослідження показує, що більшість помилок - це помилки в даних і помилок логіки, основними причинами, які викликають помилки є структурна неефективність і неправильний параметр моделі.

4.1 Характеристик помилок поглиблених вивчення

Було вибрано п'ять найпопулярніших бібліотек глибокого навчання Caffe, Keras, Tensorflow, Theano, і Torch. У той час як кожна з цих бібліотек призначена для глибокого навчання, у них різні цілі проектування. Наприклад, Tensorflow фокусується на забезпеченні низько рівневі, високо конфігуровані об'єкти, в той час як Keras прагне забезпечити високий рівень абстракцій, що приховують низько рівневі деталі. Theano і Torch зосереджені на полегшенні використання обчислень на GPU, щоб зробити глибоке вивчення більш масштабований. Таким чином, їх вивчення одночасно дозволяє порівнювати і зіставляти цілі їх проектування з помилками в їх використанні.

У даній роботі досліджується пости про ці бібліотеках на Stack Overflow. Набір даних дає нам уявлення про помилки, з якими стикаються розробники при створенні програмного забезпечення з використанням бібліотек глибоким вивченням.

Методика

Збір даних

Набори даних були знайдені у відкритій формі в мережі Інтернет. Короткий опис цих наборів даних наведено в таблиці 4.1.

Таблиця 4.1 – Зведення використаного в дослідженні набору даних

Бібліотеки	Stack Overflow	
	Записи	Помилки
Caffe	183	35
Keras	567	162
Tensorflow	1558	166
Theano	231	27
Torch	177	25
Всього	2716	415

Для вивчення помилок використання бібліотек глибокого навчання були використані дані з Stack Overflow. Це відомий сайт питань і відповідей, на якому розробники обговорюють проблеми розробки програмного забезпечення. Процес збору даних складається з двох етапів.

На першому етапі вибираються набори даних, присвячені бібліотекам глибокого навчання. Дані набори даних є у вільному доступі. Були вибрані п'ять бібліотек глибокого навчання: Caffe, Keras, Tensorflow, Theano і Torch. Це п'ять найбільш обговорюваних бібліотек глибокого навчання на Stack Overflow. Додатково були відфільтровані повідомлення, що не містять

вихідного коду, оскільки повідомлення про помилки зазвичай містять фрагменти коду. Крім того, було скорочено кількість постів, вибравши пости, чиї оцінки, розраховані як різниця між кількістю позитивних і негативних голосів, були більше 5, щоб зосередитися на високоякісних постах і зберегти керованість вручну.

На другому етапі були прочитані всі пости, щоб визначити, які з них містять помилки. Для кожного посту дивилися питання і найбільш популярну відповідь. Якщо найбільш прийнятним відповіддю було виправлення використання API-інтерфейсів глибокого навчання в питанні, то цей пост говорить про помилки глибокого навчання. Після цього кроку виявили 35, 162, 166, 27 і 25 помилок для Caffe, Keras, Tensorflow, Theano і Torch відповідно.

Класифікація корінних причин появи помилок

Відсутність сумісності Inter API

Основна причина цих помилок - неузгодженість поєднання двох різних типів бібліотек. Наприклад, користувач не може безпосередньо використовувати функцію Numpy в Keras, тому що ні бекенда Tensorflow, ні бекенда Theano Keras не мають реалізації функцій Numpy.

Відсутність перевірки типу

Цей вид помилок пов'язаний з проблемою невідповідності типів при виклику методів API. Ці помилки зазвичай пов'язані з використанням невірного типу параметрів в API.

Зміна API

Причина цих помилок - випуск нових версій бібліотек машинного навчання з несумісними API. Іншими словами, помилка виникає, коли нова версія API не має зворотної сумісності з попередньою версією. Наприклад, користувач оновлює нову версію бібліотеки машинного навчання з новим

синтаксисом API; проте користувач не змінює свій код для відповідності нової версії, що призводить до помилки зміни API.

Неправильне використання API

Подібні помилки часто виникають, коли користувачі використовують API машинного навчання, не розуміючи повністю. Відсутні умови можуть бути одним з видів неправильного використання API, і ця помилка виникає, коли використання не відповідає обмеженням API для забезпечення певних необхідних умов. Збій - головний ефект цих помилок.

Плутанина з обчислювальною моделлю

Ці помилки відбуваються, коли користувач не розуміє функції API глибокого навчання, що призводить до неправильного використання моделі обчислень, прийнятої бібліотекою глибокого навчання. Наприклад, користувач плутається між побудовою графіка і етапом оцінки.

Неправильні параметри або структура моделі (IPS)

IPS викликає проблеми з побудовою моделі машинного навчання, наприклад неправильна структура моделі або використання невідповідних параметрів. IPS - поширена помилка в програмному забезпеченні для машинного навчання через відсутність у користувачів знань про та незрозумілості моделей машинного навчання. Цей вид помилок викликає некоректність роботи; таким чином, ефект цієї помилки - збій.

Інше

Ці помилки не пов'язані з ПО для глибокого навчання. Іншими словами, ці помилки в основному пов'язані з помилками в процесі розробки, такими як неправильний синтаксис.

Неефективність структури (SI)

SI викликає проблеми, пов'язані зі стадією моделювання в програмному забезпеченні глибокого навчання, такому як IPS; проте SI призводить до поганої продуктивності, а IPS призводить до збою.

Невирівняний тензор (UT)

Ці помилки часто виникають на етапі побудови графа обчислень. Коли користувач будує граф обчислень в процесі глибокого навчання, він повинен надати правильні вхідні дані, які задовольняють вхідним специфікаціям API глибокого навчання; однак багато користувачів не знають специфікації API або неправильно розуміють підпис API, що призводить до помилок UT.

Неправильна документація

До цих помилок призводить невірна інформація в документації бібліотеки. Користувачі бібліотеки глибокого навчання можуть зіткнутися з такими помилками, якщо прочитають неправильне визначення або неправильне використання API глибокого навчання з документації.

Класифікація наслідків помилок

Погана продуктивність

Погана продуктивність або низька продуктивність - один з поширених ефектів програмного забезпечення для машинного навчання. Більш того, основними причинами цього ефекту є SI або CCM, які пов'язані з побудовою моделі. Незважаючи на те, що розробники можуть правильно використовувати бібліотеки машинного навчання, вони як і раніше стикаються з проблемами побудови моделей, оскільки API в цих бібліотеках є абстрактними.

Збій

Збій - найбільш частий ефект машинного навчання. Фактично, будь-які помилки можуть привести до збою. Ознакою збою є те, що програма перестає працювати і виводить повідомлення про помилку.

Пошкодження даних

Ця помилка виникає, коли дані пошкоджені при передачі по мережі. Цей ефект є наслідком неправильного розуміння алгоритмів або API глибокого навчання. Коли відбувається пошкодження даних, користувач отримує несподівані вихідні дані.

Зависання

Ефект зависання виникає, коли програмне забезпечення глибокого навчання перестає реагувати на введення. Повільне обладнання або невідповідний алгоритм машинного навчання можуть привести до зависання. Ознакою зависання є те, що програмне забезпечення працює протягом тривалого періоду часу, не забезпечуючи бажаний результат.

Неправильна функціональність

Цей ефект виникає, коли програмне забезпечення поводиться несподіваним чином без будь-яких помилок / попереджень часу виконання або компіляції. Це включає неправильний формат виведення, небажану роботу шарів моделі і т. Д.

Мало пам'ять

Програмне забезпечення машинного навчання часто зупиняється через недоступність ресурсів пам'яті. Це може бути викликано або неправильної структурою моделі, або недоліком обчислювальних ресурсів для навчання конкретної моделі.

Часті помилки

Нормалізований розподіл типів помилок в даних Stack Overflow показано на рисунку 4.1. Розподіл помилок і дані Stack Overflow в таблиці 4.2 показують наявність різних типів помилок.

Помилки даних

З рисунка 4.1 ми бачимо, що серед типів помилок помилки даних часто зустрічаються (26%) у всіх бібліотеках. У вивчених даних Stack Overflow бачимо 30% записів в Tensorflow , 24% записів в Keras , 36% записів в Torch, 35% записів в Theano і 9% повідомлень в Caffe з помилками даних. Помилки даних в основному виникають через відсутність обробки даних, таких як проектування функцій, перевірка даних, перетасування даних і т. д.

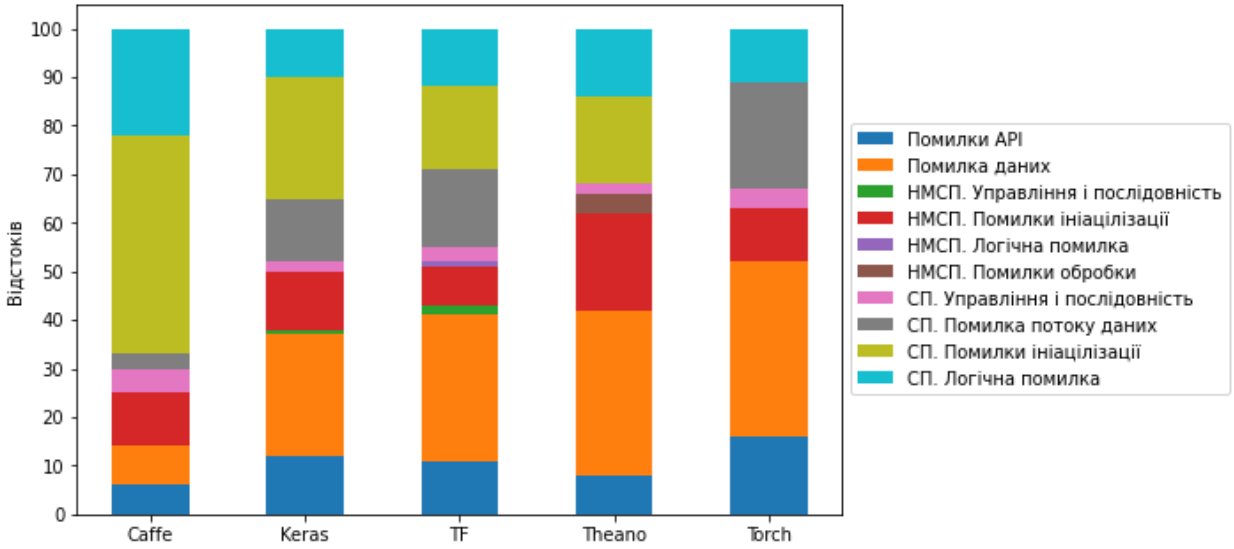

Рисунок 4.1 – Розподіл типів помилок в StackOverflow

Цю помилку складно виправити, просто глянувши на запис про помилку. Важко визначити точну причину помилки, через яку розробник розмістив питання на Stack Overflow.

Великий відсоток помилок даних вказує на те, що труднощі, пов'язані з попередньою обробкою даних, досить часто зустрічаються в програмах для машинного навчання.

Помилки структурної логіки

Другий великий тип помилки - це помилка структурної логіки в переповненні стека. В Caffe більше помилок структурної логіки в Stack Overflow в порівнянні з іншими бібліотеками. В інших бібліотеках також є значна частина помилок структурної логіки в діапазоні від 0% до 27%.

Помилки API

У бібліотеках машинного навчання зміни API іноді порушують весь виробничий код. Неявні залежності між бібліотеками викликають проблеми, коли в одній з бібліотек відбуваються серйозні зміни. Наприклад, коли Numpy оновлює Tensorflow , програмне забезпечення Keras може вийти з ладу. Keras часто використовує Tensorflow або Theano як бекенда, тому оновлення

Tensorflow або Theano може привести до збою програмного забезпечення, розробленого з використанням Keras. Помилки API частіше виникають в Keras і Tensorflow , як показано на рисунку 4.1. Більш 81% помилок API пов'язані з Keras і Tensorflow.

P-значення (англ. P-value) - величина, яка використовується при тестуванні статистичних гіпотез. Фактично це ймовірність помилки при відхиленні нульової гіпотези (помилки першого роду).

Таблиця 4.2 – Статистика типів помилок в StackOverflow і Github

	Caffe	*Keras*	*Tensorflow*	*Theano*	*Torch*	*P Value*
Помилки API	6%	11%	11%	7%	16%	0.3207
Помилка даних	9%	24%	30%	35%	36%	0.3901
НМСП. Управління і послідовність	0%	0%	0%	4%	0%	0.3056
НМСП. Помилки ініціалізації	0%	1%	1%	0%	0%	0.7655
НМСП. Логічна помилка	11%	13%	8%	25%	12%	0.0109
НМСП. Помилки обробки	0%	0%	1%	0%	0%	0.2323
СП. Управління і послідовність	6%	2%	4%	4%	8%	1
СП. Помилка потоку даних	3%	13%	15%	0%	4%	0.2873
СП. Помилки ініціалізації	0%	1%	8%	0%	20%	0.8446
СП. Логічна помилка	42%	27%	18%	18%	0%	0.3442
СП. Помилки обробки	23%	8%	4%	7%	4%	0.8535

Основні причини виникнення помилок

Нормалізоване розподіл причин в фрагментах коду Stack Overflow показано на рисунку 4.2. Дані в таблиці 4.2 показують наявність різних

категорій основних причин в Stack Overflow і представляють значення P, що показує схожість розподілів з використанням t-тесту.

Неправильний параметр моделі (IPS)

IPS призводить до помилок, які викликають збій програми під час виконання і виконання не вдається. У Tensorflow і Theano IPS призводить до інших основних причин виникнення помилок, що становлять 26% і 26% від загальної частки основних причин, відповідно.

Структурна неефективність (SI)

Помилки SI не викликають збоїв програми. Ці помилки часто призводять до неоптимальної продуктивності моделі глибокого навчання. Ці помилки більше пов'язані з QoS або нефункціональними вимогами. Наприклад, програміст намагається навчити модель розпізнавати рукописні цифри, але точність не поліпшується і залишається постійною 2-10 епох.

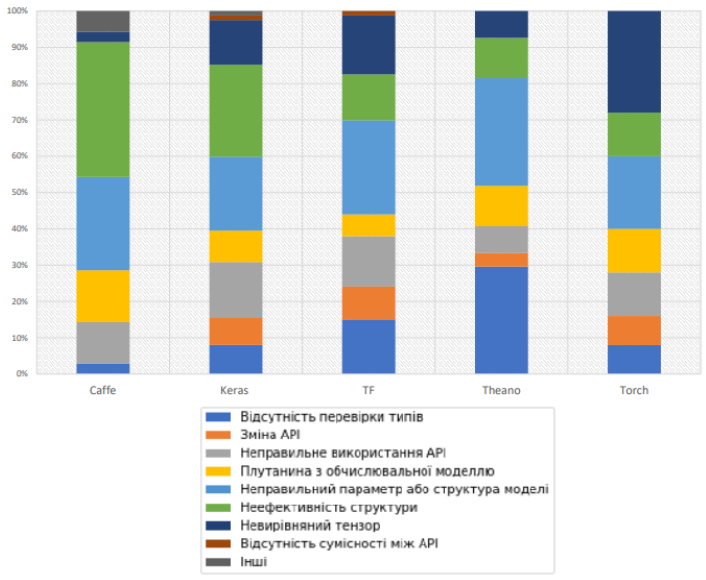

Рисунок 4.2 – Класифікація причин Stack Overflow

Не вирівняний тензор (UT)

У машинному навчанні тензорні вимірювання важливі для успішної побудови моделі. Tensorflow , Keras , Torch , Theano , Caffe мають 16%, 12%, 28%, 7% і 3% помилок через UT відповідно.

Відсутність перевірки типу

Більшість бібліотек машинного навчання написано на Python. Через динамічної природи Python проблема відсутності перевірки типів сильно відчувається в цих бібліотеках. Відсутність перевірки типів призводить до 30% помилок в Theano , 8% помилок в Keras і 15% помилок в Tensorflow.

Зміна API

У бібліотеках глибокого навчання зміна API має тенденцію мати рішучий ефект. Ці бібліотеки взаємозалежні. Отже, зміна API в одній бібліотеці порушує роботу інших бібліотек.

Зв'язок основної причини з типом помилки

З рисунку 4.3 видно, що більшість помилок, не пов'язаних з моделлю, викликані неправильним використанням API (6-100%). Помилки структурної ініціалізації, які не є моделями, і помилки структурної обробки, які не є моделями, викликані неправильним використанням API в 100% випадків в вивчених даних. Цікаво, що в помилках зміна API грає життєво важливу роль (68%) у порівнянні з неправильним використанням API (20%); однак помилки, пов'язані з моделлю, більш уразливі для корінних причин IPS і SI. З рисунку 4.3 видно, що помилка структурного контролю та послідовності, помилка структурного потоку даних, помилка структурної ініціалізації, помилка структурної логіки, помилка структурної обробки, які пов'язані з моделлю, викликані SI 31%, 3%, 10%, 33% і 53. % раз відповідно і викликано IPS 62%, 59%, 40%, 36%, 24% раз відповідно.

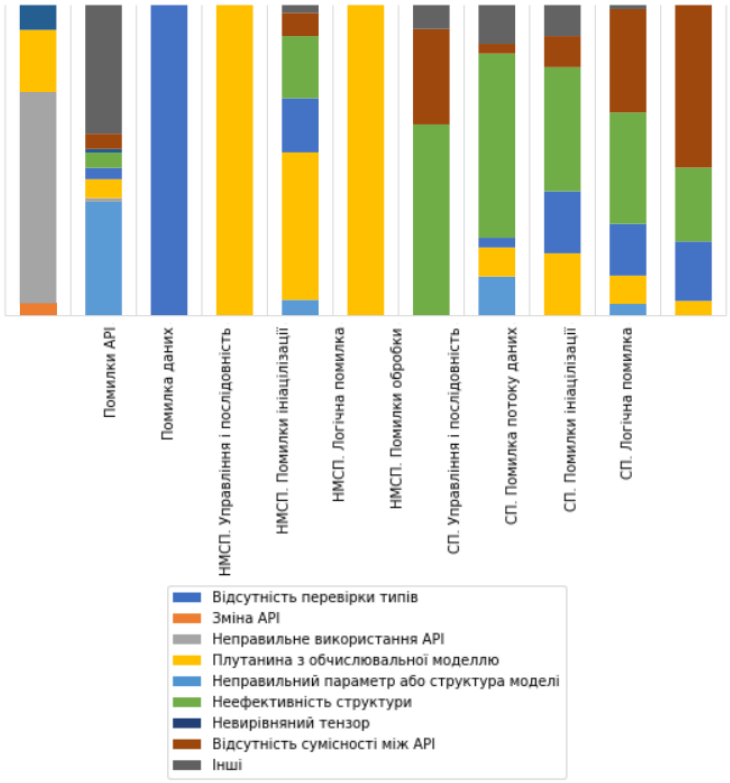

Рисунок 4.3 – Зв'язок між кореневими причинами і типами помилок

Вплив помилок

Нормалізований розподіл ефектів Stack Overflow показано на рисунку 4.4. Дані в таблиці 4.3 показують наявність різних видів ефектів для бібліотек машинного навчання.

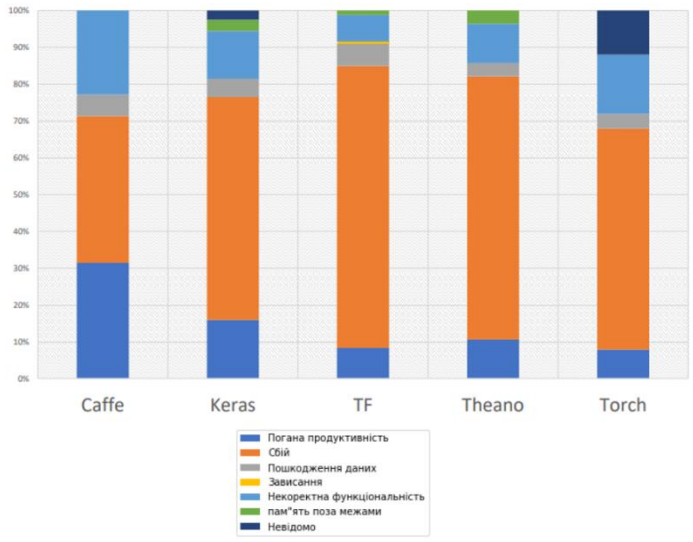

Рисунок 4.4 – Розподіл ефектів помилок Stack Overflow

Збій

Аналіз показує, що найбільш серйозним наслідком помилок є збій. У глибокому навчанні помилки в основному приводять до повного збою програми. У всіх бібліотеках найбільший вплив робить збій в діапазоні від 40% до 77%, як показано на рисунку 2.4.

Погана продуктивність

Низька продуктивність часто викликає занепокоєння у розробників програмного забезпечення для глибокого навчання. Навіть якщо модель навчається успішно, на етапі оцінки або прогнозування модель може давати дуже низьку точність при класифікації цільових класів.

Неправильне функціональність

Неправильна функціональність виникає, коли поведінка програмного забезпечення під час виконання призводить до якогось незрозумілого результату, якого не можна очікувати від логічної організації моделі або попереднього досвіду розробника.

Наприклад, в наступному фрагменті коду користувач хоче перетворити зображення в масив 28 * 28 Numpy ; проте на виході виходить чорне зображення (рис. 4.5).

```
with tf.Session() as sess:
    first_image = mnist.train.images[0]
    first_image = np.array(first_image, dtype='uint8')
    pixels = first_image.reshape((28, 28))
    plt.imshow(pixels, cmap='gray')
```

Рисунок 4.5 – Приклад використання бібліотеки Numpy

Користувач отримав неправильний висновок через перетворення масиву з плаваючою комою в uint8, який перетворює всі пікселі в 0, якщо вони менше 1. Щоб вирішити проблему, користувач може помножити масив на 255, як пропонується у відповіді. Theano має більш високий відсоток повідомлень про

проблеми з неправильною функціональністю в порівнянні з поганою продуктивністю.

Складні етапи навчання

Підготовка даних

З рисунку 4.6 видно, що більшість помилок в програмуванні машинного навчання відбувається на етапі підготовки даних.

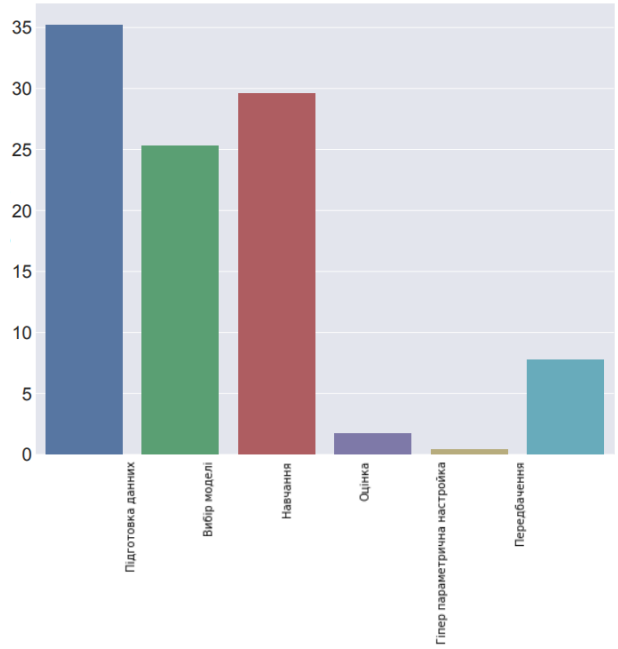

Рисунок 4.6 – Помилки на етапах конвеєра глибокого вивчення

Етап навчання

Наступним етапом, схильним до помилок, є етап навчання. Більшість помилок, пов'язаних з IPS і SI, виникає на етапі навчання.

Вибір моделі

Вибір моделі - третій етап помилки. На етапі вибору моделі будується модель і вибирається правильний алгоритм. Основними причинами помилок на цьому етапі є IPS, SI і UT.

Спільність помилок

Основна гіпотеза полягала в тому, що бібліотеки будуть сильно корельовані на основі розподілу помилок, оскільки вони виконують схожі завдання.

Аналіз підтверджує цю гіпотезу, як показано на рисунку 4.7. Видно , що бібліотеки мають сильний коефіцієнт кореляції, близький до 1. Дивно, але Caffe показала дуже слабку кореляцію з іншими бібліотеками з точки зору типу помилки.

Було виявлено антипатерн шляхом більш глибокого аналізу коду помилок Stack Overflow для подальшого дослідження сильної кореляції Tensorflow і Keras , а також слабкою кореляції Torch і Caffe. Виявлено антипатерн : безперервне старіння, програмування методом вирізання і вставки, мертвий код, золотий молоток, затичка на введення даних, управління грибами, код спагеті. Ця класифікація взята з [31]. Розподіл різних антипатерн по бібліотеках показано на рисунку 4.8, але , що в Tensorflow і Keras 40% антипатерн - це затичка на введення даних. З іншого боку, в Torch 40% помилок виникає через антипатерн «вирізати і вставити». Tensorflow і Keras мають майже однакову розподіл в Безперервне старіння і Мертвий код. Це показує, що сильна кореляція між розподілом помилок в Tensorflow і Keras може бути пояснена схожістю загальних антипатерн для цих двох бібліотек. Слабка кореляція між розподілом помилок Torch і Caffe може бути результатом різного розподілу антипатерн між цими двома бібліотеками.

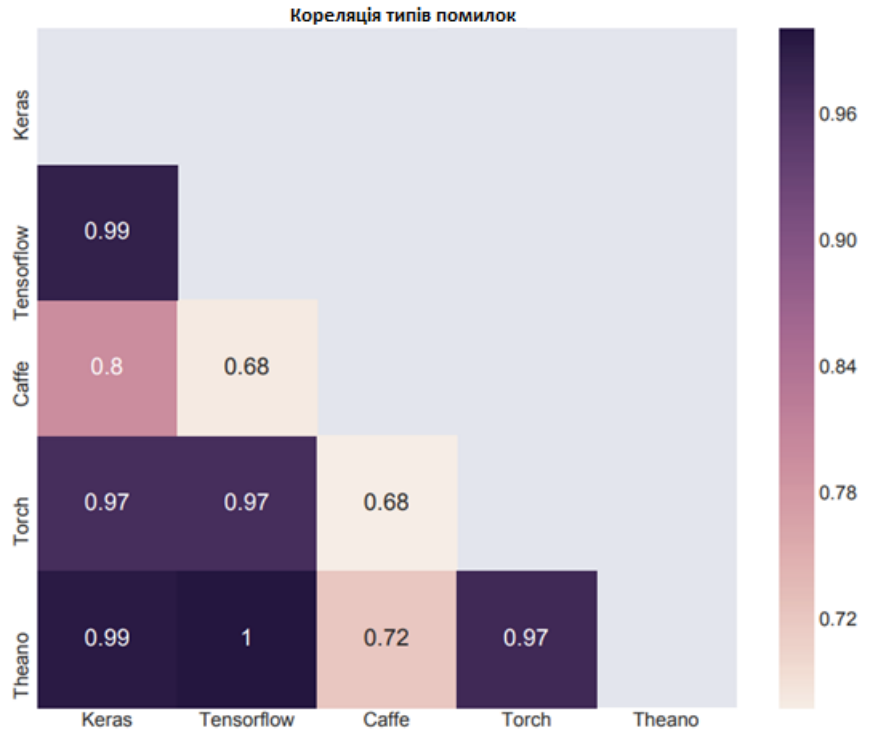

Рисунок 4.7 – Кореляція типів помилок серед бібліотек

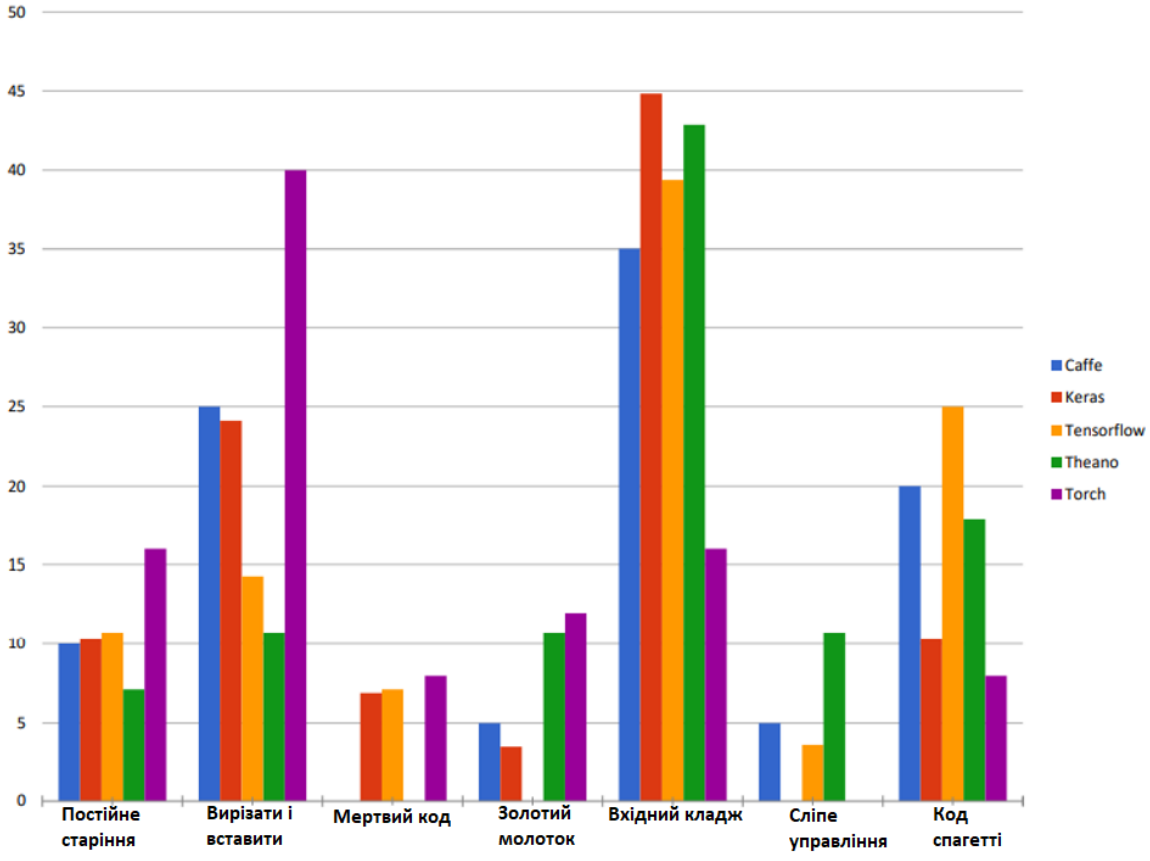

Рисунок 4.8 – Розподіл різних анти параметрів

Еволюція помилок

Зростання числа помилок структурної логіки

З 2015 по 2018 помилки структурної логіки в Caffe складають відповідно 30%, 32%, 67%, 100%, що вказує на те, що помилки структурної логіки все більше обговорюються розробниками з 2015 року. Очікувано , що з 2015 року машинне навчання стало залучати все більшу увагу. розробники почали використовувати бібліотеки машинного навчання для написання програмного забезпечення.

Зменшення кількість помилок в даних

В Torch Data помилки залишалися майже незмінними, підтримуючи близько 50% помилок, що обговорювалися в 2016-2018 роках (рис.4.9). У Keras помилки даних повільно зменшувалися з 27% до 15% з 2015 року. У Tensorflow помилки даних повільно зменшувалися з 30% до 10% з тих пір. 2015-2018 рр. У двох інших бібліотеках кількість помилок даних також поступово зменшувалася, наближаючись до нуля. Можливою причиною цієї тенденції є розвиток популярних спеціалізованих бібліотек даних, таких як pandas , які дозволяють проводити дослідний аналіз даних для кращого розуміння властивостей даних. Крім того, використання типу даних Tensor , що має інформацію про тип і форму, допомагає позбутися від деяких помилок даних. Додаткова підтримка верифікації в цих бібліотеках допоможе позбутися від цих помилок.

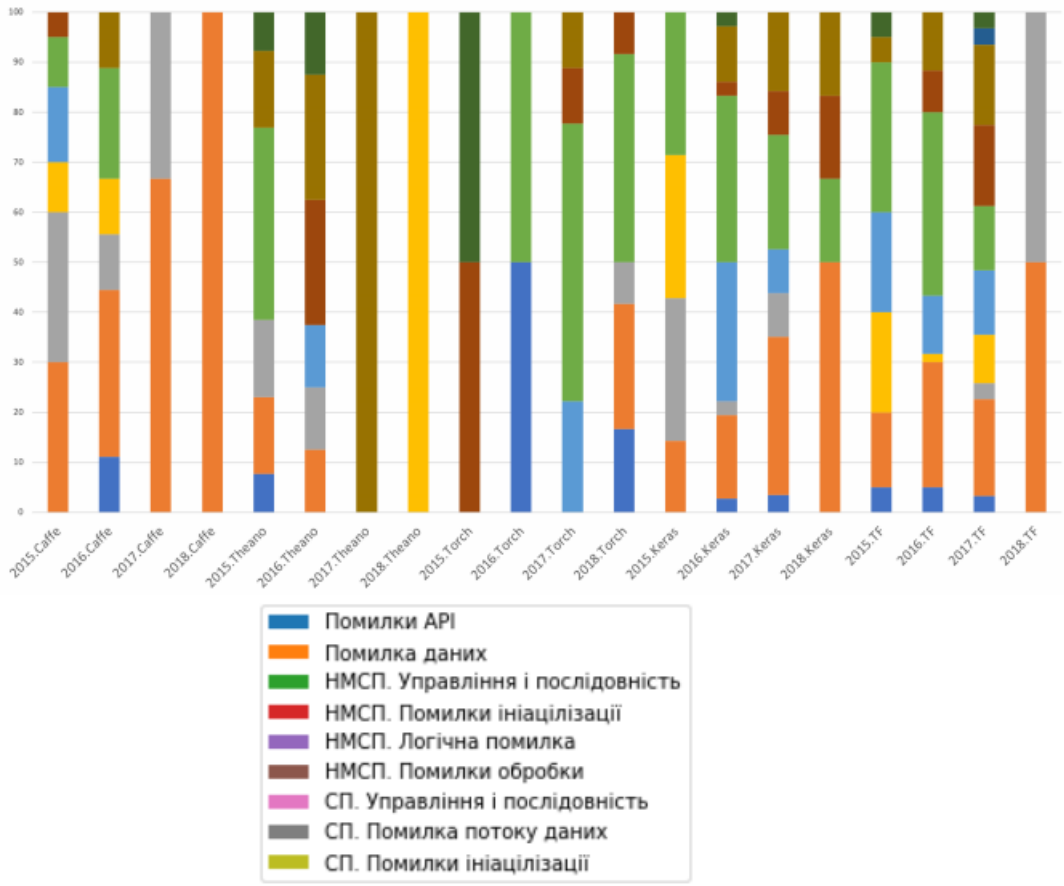

Рисунок 4.9 – Терміни еволюції помилок

Таблиця 4.3 – Статистика корінних причин появи помилок

	Caffe	*Keras*	*Tensorflow*	*Theano*	*Torch*	*P Value*
Відсутність сумісності між API	0%	1%	1%	0%	0%	0.1411
Відсутність перевірки типів	3%	8%	15%	30%	8%	0.9717
Зміна API	0%	7%	9%	4%	8%	0.2485
Неправильне використання API	11%	15%	14%	7%	12%	0.003
Плутанина з моделлю обчислень	14%	9%	6%	11%	12%	0.7839

Неправильний параметр моделі або структури	26%	21%	26%	30%	20%	0.5040
Друге	0%	0%	0%	0%	0%	0.3466
Недостатність структури	37%	26%	13%	11%	12%	0.7170
Невирівняний тензор	3%	12%	16%	7%	28%	0.7541
Неправильна документація	6%	1%	0%	0%	0%	0.3402

Таблиця 4.4 – Вплив помилок в Stack Overflow

	Caffe	*Keras*	*Tensorflow*	*Theano*	*Torch*	*P Value*
Погана продуктивність	31%	16%	8%	11%	8%	0.9152
Збій	40%	61%	77%	70%	60%	0.7812
Пошкодження даних	6%	5%	6%	4%	4%	0.948
Зависання	0%	0%	1%	0%	0%	0.3466
Некоректна функціональність	23%	13%	7%	11%	16%	0.5418
Пам'ять поза межами	0%	3%	1%	4%	0%	0.0844
Невідомо	0%	2%	0%	0%	12%	0.8419

4.2 Виправлення закономірностей і проблем

Доступність великих даних привела до появи глибоких нейронних мереж (DNN). DNN складається з набору шарів. Кожен рівень містить набір вузлів, які збирають вхідні дані з попереднього рівня і подають вихідні дані вузлів наступного рівня через набір зважених ребер. Ці ваги коректуються з використанням прикладів, які називаються навчальними даними, і встановлюються на значення, які мінімізують різницю між фактичними вихідними даними DNN і очікуваними вихідними даними, виміряними з використанням цільової функції, званої функцією втрат. Доступність великих даних дозволила точно налаштувати ваги для багатошарових DNN. Таким чином, багато програмних систем зазвичай використовують DNN.

Суттєвою проблемою створення програмного забезпечення, яке використовує DNN, є наявність помилок [32]. Визначили типи помилок, основні причини та їх наслідки в бібліотеці Tensorflow для DNN. У [33] вивчили ще більший набір бібліотек, включаючи Caffe, Keras, Tensorflow, Theano і Torch, щоб визначити характеристики помилок. У той час як попередні роботи представляють собою первинне дослідження шаблонів виправлення для Tensorflow, ця робота сфокусуватись на характеристиках виправлення. Оскільки відновлення програмного забезпечення, що використовує DNN, є очевидною потребою в системній інженерії, де автоматизовані інструменти можуть бути дуже корисні, повне розуміння проблем відновлення і шаблонів, які використовуються при ручному виправленні помилок в DNN, має вирішальне значення.

Методика

Набір даних

Дане дослідження спирається на набір даних про помилки, підготовлений в [33] для збору і підготовки набору даних про виправлення помилок. Набір даних про помилки містить 415 помилок з Stack Overflow для 5 різних бібліотек глибокого навчання, як показано в Таблиці 4.5.

Таблиця 4.5 – Зведені дані щодо виправлення помилок

Бібліотеки	Помилки	Виправлення
Caffe	35	27
Keras	162	143
Tensorflow	166	118
Theano	27	15
Torch	25	17
Всього	415	320

Збір виправлень помилок Stack Overflow

Щоб зібрати виправлення помилок в наборі даних помилок Stack Overflow, було вивчено відповіді які мають найбільшу оцінку. Якщо в записі немає коду, але описується, що потрібно виправити в вихідної помилку, ми також розглядаємо їх як виправлення. Якщо запис про помилку не має прийнятого відповіді, але має відповідь оцінку > = 5 балами, ми розглядаємо їх як виправлення, так як оцінка 5 вважається прийнятною метрикою якості в попередніх роботах [33]. Дотримуючись цієї методології, було знайдено 320 виправлень для 320 записів.

Виправлення помилки класифікації шаблонів

Функція втрати

Ця група виправлень заснована на додаванні, видаленні або оновленні функції втрат під час навчання. Функція втрат - це ключовий параметр, який допомагає процесу навчання виявити відхилення від засвоєних і реальних прикладів. Для різних типів завдань потрібні різні функції втрат, наприклад, крос-ентропійні втрати широко використовуються в задачах класифікації, тоді як середньоквадратичні втрати помилок в основному використовуються для задач, заснованих на регресії. Деякі проблеми вимагають налаштування функції втрат для кращого результату навчання.

Підключення до мережі

Ця група виправлень змінює зв'язок між вузлами в глибока нейронна мережа (DNN). DNN - це граф, де ребра - це ваги і зміщення, а вузли - це елементи кожного шару. Наприклад, в щільному шарі ребра ваги повністю пов'язані з наступним шаром, і розмір шару визначає кількість вузлів, які будуть доступні в цьому шарі. Ті виправлення помилок, які змінюють конфігурацію цих сполучень для поліпшення результатів, класифікуються в цій категорії. Зміни включають зміну ваги, видалення ребер шляхом обрізки мережі, додавання зворотного поширення і т. д.

Додавання рівня

У будь-якому завданні, заснованої на класифікації, в моделі буде як мінімум два прошарки: вхідний і вихідний. Щоб вивчити особливості введення DNN потрібно більше проміжних шарів (званих прихованими). Ця група виправлень додає додаткові рівні DNN для підвищення продуктивності. Додані шари можуть бути щільними, коли два послідовних шару повністю з'єднані, згорнутим шаром, де при вході застосована функція згортки, шаром виключення для зменшення перенавчання і т.д.

Розмір шару

Ці виправлення змінюють розміри шарів, щоб зробити їх сумісними з сусідніми шарами і введенням.

Вимірювання даних

Виправлення, пов'язане з вимірюванням даних, аналогічно вимірюванню шару, але воно пов'язане з вхідними даними, а не з шарами DNN. Розмір даних повинен бути узгоджений з DNN. Цей тип виправлення найчастіше потрібно, коли вхідні розміри DNN і вимірювання даних не збігаються.

Метрика точності

Щоб виміряти правильність DNN, показник точності є одним з ключових параметрів, які необхідно налаштувати. Тип проблеми має величезний вплив на тип використовуваної метрики точності, наприклад, проблеми класифікації оцінюються з використанням точності класифікації, оцінки F1 або матриці плутанини, але ці метрики НЕ підходять для оцінки моделі, заснованої на регресії, де логарифмічні втрати більше підходять.

Тип даних

Ця група виправлень змінює тип вхідних даних, щоб відповідати очікуванням DNN.

Активація

Функція активації для вузла в шарі DNN зіставляє вхідні дані з вихідними. Ця група виправлень змінює функцію активації, яка використовується в шарі, щоб краще відповідати проблемі.

Ітерації

Ця група виправлень регулює кількість запусків тренувального процесу. Зазвичай це робиться для підвищення точності або зменшення перенавчання. Ці виправлення включають зміна розміру пакета або епохи. У деяких випадках розробники додають цикл всього навчального процесу.

Версія

Бібліотеки DNN швидко розвиваються, і ряд випусків не мають зворотної сумісності, що порушує код. Ця група виправлень адаптує код для роботи з новою версією бібліотеки DNN.

Перетворення даних

Суперечки даних відносяться до зміни форми даних без зміни їх цілі. Зазвичай це робиться для виправлення даних для подальших операцій. Ця група виправлень додає обробку даних для виправлення DNN, наприклад шляхом зсуву даних, перемішування і т.д.

Монітор

Виправлення в цій категорії додають код для діагностики в процесі навчання, зазвичай для друку статистики навчання. Ця група виправлень не усуває помилку в коді, але допомагає локалізувати помилку.

Оптимізатор

Ця група виправлень змінює алгоритми оптимізації, використовувані моделлю DNN. Алгоритм оптимізації, що залежить від проблеми, визначає ітераційний процес, що виконується для підвищення точності моделі DNN.

Зміна нейронної архітектури

Ця група виправлень, по суті, переробила DNN модель, так як вихідна модель була непридатна.

Шаблони виправлень для різних типів помилок

Щоб дізнатися, чи відрізняються шаблони виправлення помилок для різних типів помилок, було проаналізовано кореляцію між типами помилок в наборі даних, представленому в [33], і патернами виправлення помилок, з використанням розподілу помилок і відповідних їм виправлень. Розподіл патернів виправлення помилок за різними типами помилок в Stack Overflow показано на рисунках 4.10 і 4.11 відповідно. Горизонтальна і вертикальна осі описують різні типи помилок з [33] і відсоток різних шаблонів виправлень, необхідних для виправлення цих помилок, відповідно.

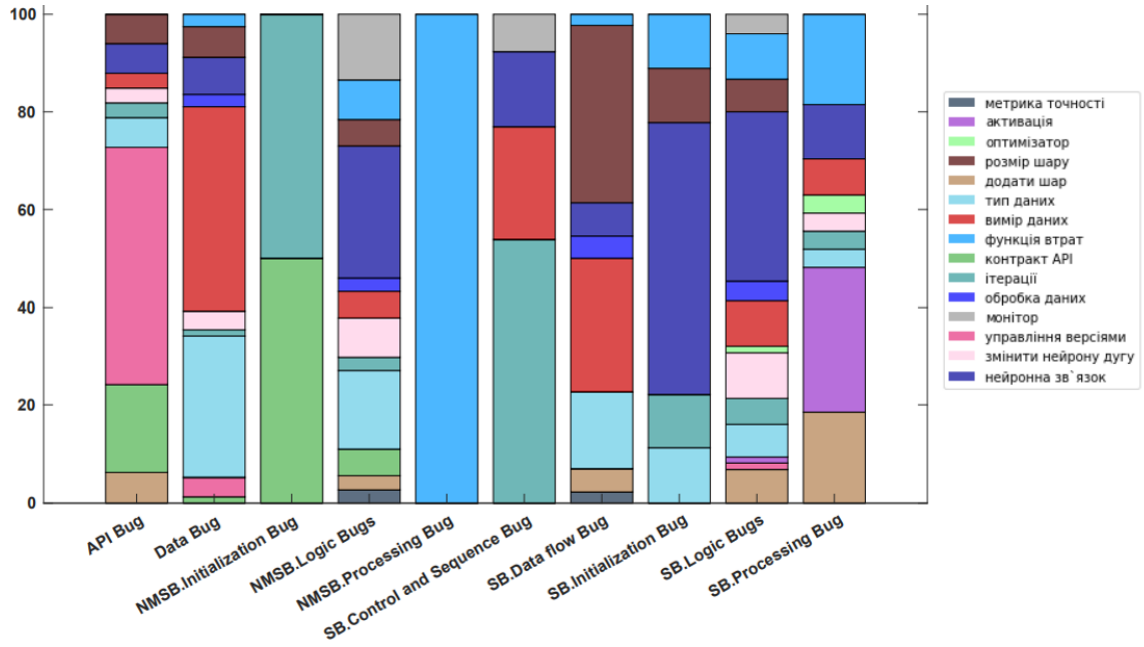

Рисунок 4.10 – Розподіл шаблонів для виправлення Stack Overflow різних типів помилок

Виправлення специфікацій API включає зміну контрактів API через управління версіями API і підтримки міжбібліотечних операцій в моделі. Виправлення специфікацій API необхідно з наступних причин.

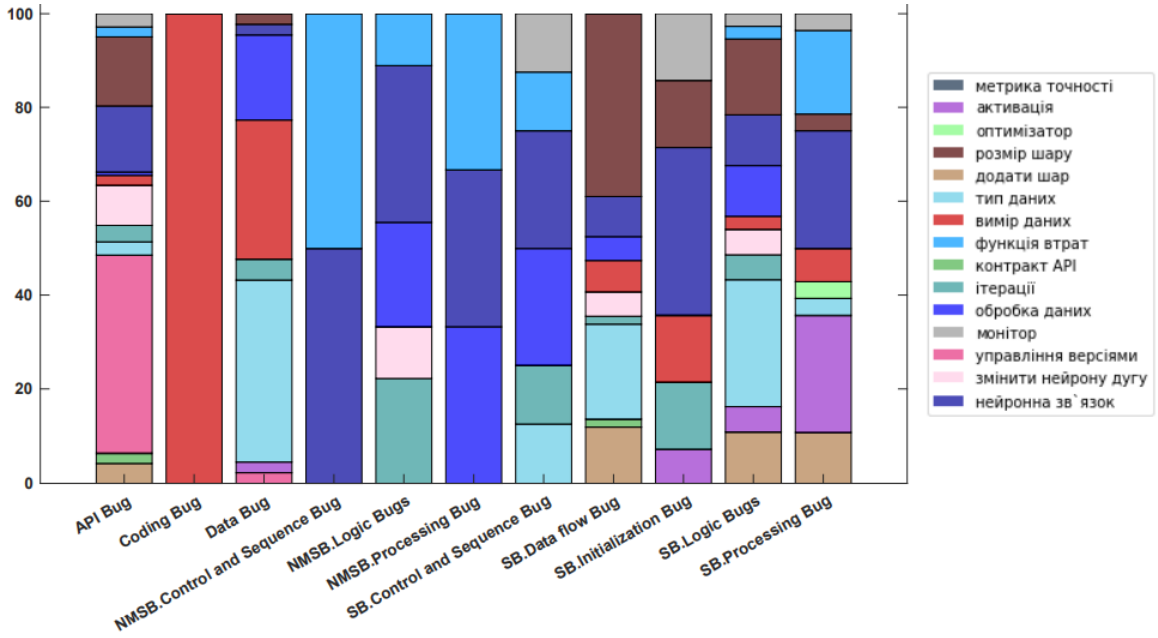

Рисунок 4.11 – Розподіл шаблонів для виправлення Stack Overflow різних типів помилок

Зміна характеристик у зв'язку з оновленням версії.

20 виправлень в Stack Overflow включають зміна специфікацій, які потрібні у зв'язку зі зміною версії бібліотеки. Зміни під час оновлення версії бібліотеки включають такі зміни: зміна повних імен методів, зміна сигнатури API і зміна імовірнісного поведінки API. Хоча виправлення, пов'язані зі зміною повних імен методів і зміною сигнатури API, є добре вивченими проблемами, виправлення, пов'язані зі зміною імовірнісного поведінки API, складні і відрізняються від традиційних змін API. Локалізація цих помилок ускладнена через відсутність складних інструментів імовірнісного аналізу для машинного навчання.

Зміна специфікації для підтримки бібліотеки. Ці помилки виникають через схожі припущення про поведінку і специфікаціях різних API в різних бібліотеках. Для виправлення цих помилок потрібен досвід роботи з обома бібліотеками.

З рисунків 4.10 і 4.11 можна виявити, що виправлення вимірювання даних є найбільш помітним шаблоном (41,77%) для виправлення помилок даних в Stack Overflow. Це говорить про те, що для виправлення помилок даних основні зміни пов'язані з вимірами даних. Це відбувається тому, що розмір даних дуже важливий для правильної роботи моделі DNN.

Для виправлення логічних помилок найбільш поширеною практикою є зміна мережевого підключення (~ 27,03% в Stack Overflow). Беручи до уваги, що значна кількість помилок потоку даних можна виправити, змінивши розмір шару (~ 36,36% в Stack Overflow). Ці спостереження дають нам інтуїцію, що для виправлення різних типів помилок можуть знадобитися унікальні технічні підходи.

Введення помилок через виправлення

Щоб відповісти на питання чи призводить виправлення помилки DNN до нової помилку, було проаналізовано випадково вибрані виправлення

з Stack Overflow , щоб зрозуміти, чи може виправлення помилки привести до появи нової помилки. Були прочитані відповіді на відповіді, вибрані за допомогою критеріїв фільтрації. Було виявлено, що 29% виправлень в випадково обраному наборі даних вносять як мінімум одну нову помилку в коді. Тут нова помилка вказує на те, що вихідна помилка була виправлена опублікованими рішенням; проте рішення представляє нову помилку, яка відрізняється від вихідної помилки. Цей результат показує, що більшість введених помилок відносяться до нових типів, і їх поведінка повністю відрізняється від поведінки батьківських помилок. Також виявлено, що збій (55,8%) є найбільш частим впливом цих нових помилок, і більшість цих помилок пов'язано з помилкою API (37,9%), а найбільш поширеною основною причиною цих помилок є зміна API (34,5%), Який включає зміну або підписи API, або повного імені API. 44,8% і 34,5% нових помилок пов'язані з Keras і Tensorflow. Виправлення помилок, пов'язаних з Caffe, Theano і Torch, вносять 10,34%, 3,45% і 6,90% нових помилок відповідно.

Проблеми в виправленні помилок

DNN Повторне використання

Навчання DNN-моделям може бути дорогим, так як вимагає складних обчислювальних ресурсів і великого обсягу помічених даних, які можуть бути недоступні. Це призвело до повторного використання DNN-моделей, що створює унікальні проблеми, такі як атака чорного ходу, введення зміщення і невідповідність між намірами встановленою DNN-моделі і намірами розробника.

```
base_model = ResNet50(input_shape=(224, 224, 3),
include_top=False,weights='imagenet',pooling='avg')
+ x=base_model.output
+ x = Dense(512, activation='relu')(x) #add new layer
+ x = Dropout(0.5)(x) #add new layer
+ x = Dense(512, activation='relu')(x) #add new layer
+ x = Dropout(0.5)(x)
```

У наведеному вище прикладі з поста Stack Overflow #4922644714 розробник хоче побудувати визначену структуру DNN моделі ResNet50, використовуючи набір даних про рак. Навчена мережа призводить до перепідготовки, так як розробник не знав про структуру повторно використовуваної моделі і потребував зміні мережі шляхом додавання випадають і щільних шарів для зменшення ефекту перепідготовки.

Відстежувана або частково відстежувана помилка

У разі помилки падіння, загальною стратегією локалізації помилки є аналіз повідомлення про помилку. Проте, було виявлено, що локалізація помилки дуже складна в програмному забезпеченні DNN, так як помилки і збої не пов'язані між собою. Для ілюстрації розглянемо фрагмент коду з повідомлення Stack Overflow # 3347442415 нижче.

```
model = Sequential()
model.add(Dense(hidden_size, input_dim=input_size, init='uniform'))
model.add(Activation('tanh'))
...
y_pred = model.predict(X_nn)
```

Цей код видає наступний слід помилки:

```
ttributeError                              Traceback (most recent call last)
<ipython-input-17-e6d32bc0d547> in <module>()
1
----> 2 y_pred = model.predict(X_nn)
491     def predict(self, X, batch_size=128, verbose=0):
492         X = standardize_X(X)
--> 493         return self._predict_loop(self._predict, X, batch_size,
verbose)[0]
494
495     def predict_proba(self, X, batch_size=128, verbose=1):

AttributeError: 'Sequential' object has no attribute '_predict'
```

З цього повідомлення про помилку розробник може почати копатися в коді функції прогнозу і об'єкта Sequential , але проблема полягає в пропущеному кроці компіляції. У зв'язку з цим підключення моделі не ініціалізується, а помилка поширюється на роботу функції прогнозу і зупиняє процес навчання. Було виявлено, що 11 з 50 повідомлень не вказують на повідомлення про помилку, а в інших 39, 20 повідомлень мають виправлення, яке не збігається з повідомленням про помилку.

Швидкі версії

Раніше обговорювалось, що велика кількість виправлень пов'язано з швидким збільшенням версій і змінами в DNN-бібліотеках.

У таблиці 4.6 наведено кількість символів операцій, доступних для кожної версії Tensorflow , а також кількість операцій, які були видалені, перейменовані або переконфігурвати в порівнянні з попередньою версією. Було виявлено, що з версії 1.14 на версію 2.0 було змінено 26% операцій. Також було вивчено Keras v2.0, v2.1, v2.2 і v2.3, щоб зрозуміти, чи є ця проблема поширеною тільки в Tensorflow чи ні. Встановлено, що при переході від v2.0-v2.1, v2.1-v.2.2 і v2.2-v2.3 відсоток зміни операцій становить 6%, 8% і 4% відповідно. Нетривіальною проблемою при виправлені програмного забезпечення DNN є розподіл усіх поведінку API. Деякі з цих оновлень версії також змінюють розподіл усіх поведінку API, викликаючи

деякі складні помилки. Нижче представлений приклад, в якому зміна імовірнісного розподілу змінює висновок однієї і тієї ж операції з різними версіями.

Таблиця 4.6 – Зміни API Tensorflow. Зміна = # операцій змінено в порівнянні з попередньою версією

Версія	#- Символів	Зміни	Дата Випуску
v*2.0 (Beta)*	6504	2185	07.06.2019
v1.14.0	8363	59	18.06.2019
v1.13.1	3560	39	25.02.2019
v1.12.0	3314	52	01.11.2018
v1.11.0	3145	175	25.09.2018
v1.10.0	3230	N/A	07.08.2018

5 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АНАЛІЗУ РЕЗУЛЬТАТІВ ВИКОРИСТАННЯ БІБЛІОТЕК МАШИННОГО НАВЧАННЯ У РЕАЛІЗАЦІЇ ПРОЕКТІВ РОЗРОБКИ ПРОГРАМНИХ ПРОДУКТІВ

Для пришвидшення розробки та виявлення помилок у реалізації програмних продуктів була розроблена інформаційна система аналізу результатів бібліотек машинного навчання. За її допомогою можна проаналізувати код та визначити, чи містить він помилку, яка вже обговорювалася на Stack Overflow.

5.1 Набори даних

Набори даних для визначення результатів були взяті з Stack Exchange. Stack Exchange - це мережа веб-сайтів із запитаннями та відповідями на різні теми, кожна з яких охоплює певну тему, де питання, відповіді та користувачі підлягають процедурі присудження репутації [34]. Для роботи була взята база даних Stack Overflow, яка оновлюється кожного дня. Даний сервіс пропонує ввести SQL запит результатом якого є csv файл з даними, які попадають під критерії пошуку. Для моделі були взяті поля які представлені в таблиці. 5.1.

Таблиця 5.1 – Атрибути питань та відповідей

Поле	Значення	Тип
question.Id	Унікальний номер питання	int
question.CreationDate	Дата створення питання	datetime
question.ViewCount	Кількість переглядів	int
question.Tags	Теги питання	nvarchar (250)
question.Title	Заголовок питання	nvarchar (250)

question.Body	Текст питання	nvarchar (max)
question.Score	Оцінка питання	int
answer.Body	Текст відповіді	nvarchar (max)
answer.Score	Оцінка відповіді	int

Щоб отримати необхідні набори даних був створений SQL запит. Даний запит включає всі поля, які були вказані в таблиці 1, та має наступні умови:

- рейтинг відповіді більше 4;
- тіло відповіді не повинно бути пусте;
- тіло питання не повинно бути пусте;
- теги питання повинні містити назву шуканої бібліотеки;
- питання було створене після 1-го січня 2016 року.

```
SELECT
  question.Id,
  question.CreationDate,
  question.ViewCount,
  question.Tags,
  question.Title,
  question.Body,
  question.Score,
  answer.Body,
  answer.Score
FROM
  Posts question
  LEFT JOIN
    Posts answer
    ON answer.Id = question.AcceptedAnswerId
    AND answer.PostTypeId = 2
WHERE
  question.PostTypeId = 1
  AND answer.Score > 4
  AND answer.Body IS NOT NULL
```

```
  AND question.Tags LIKE '%<tag_name>%'
  AND question.Body IS NOT NULL
  AND question.CreationDate > '2016-01-01'
ORDER BY
  question.CreationDate ASC
```

Після завантаження всіх файлів, кількість записів для кожної бібліотеки представлено в таблиці 5.2. Всього, кількість записів – 6773.

Таблиця 5.2 – Кількість записів у бібліотеках

Бібліотеки	Кількість
Caffe	184
H2O	52
Keras	1559
Mahout	20
Mllib	166
Scikit-learn	1405
Tensorflow	3083
Theano	147
Torch	95
Weka	62

5.2 Реалізація моделі

Для зчитування записів з наборів даних там подальших маніпуляцій була використана бібліотека pandas [35].

Pandas - це бібліотека з відкритим кодом, що забезпечує високопродуктивні, прості у використанні структури даних та засоби аналізу даних для мови програмування Python.

Скрипт зчитування даних зображено на рисунку 5.1. Для видалення всіх пустих значень був використаний метод *dropna()*.

```
In [3]: #Caffe
        df_Caffe = pd.read_csv("./data/Caffe.csv", nrows=184)
        df_Caffe = df_Caffe.dropna()
        #H2O
        df_H2O = pd.read_csv("./data/H2O.csv", nrows=52)
        df_H2O = df_H2O.dropna()
        #Keras
        df_Keras = pd.read_csv("./data/Keras.csv", nrows=1559)
        df_Keras = df_Keras.dropna()
        # Mahout
        df_Mahout = pd.read_csv("./data/Mahout.csv", nrows=20)
        df_Mahout = df_Mahout.dropna()
        # Mllib
        df_Mllib = pd.read_csv("./data/Mllib.csv", nrows=166)
        df_Mllib = df_Mllib.dropna()
        # Scikit-learn
        df_Scikit_learn = pd.read_csv("./data/Scikit-learn.csv", nrows=1405)
        df_Scikit_learn = df_Scikit_learn.dropna()
        # Tensorflow
        df_Tensorflow = pd.read_csv("./data/Tensorflow.csv", nrows=3083)
        df_Tensorflow = df_Tensorflow.dropna()
        # Theano
        df_Theano = pd.read_csv("./data/Theano.csv", nrows=147)
        df_Theano = df_Theano.dropna()
        # Torch
        df_Torch = pd.read_csv("./data/Torch.csv", nrows=95)
        df_Torch = df_Torch.dropna()
        # Weka
        df_Wekka = pd.read_csv("./data/Weka.csv", nrows=62)
        df_Wekka = df_Wekka.dropna()
```

Рисунок 5.1 – Зчитування даних з файлів

Об’єднання всіх дата фреймів в один (рис. 5.2).

```
In [16]: df = pd.concat([df_Caffe,df_H2O, df_Keras,df_Mahout,df_Mllib,df_Scikit_learn,
                        df_Tensorflow,df_Theano,df_Torch,df_Wekka])
```

Рисунок 5.2 – Об’єднання дата фреймів

Створений дата фрейм має наступний вигляд (рис.3). Колонка “Body.1” – це тіло відповіді, “Score.1” – рейтинг відповіді.

	Id	CreationDate	ViewCount	Tags	Title	Body	Score	Body.1	Score.1
0	27890137	2015-01-11 17:44:37	9120	<c++><macos><opencv> <osx-yosemite><caffe>	Undefined symbols for architecture x86_64: for...	I got this error for <a href="http://caffe....	7	Solved! \n\n <code>cv::imread(cv::Stri...	18
1	27986339	2015-01-16 14:39:56	1847	<machine-learning> <neural-network><deep-learni...	Where can I find the label map between trained...	everyone, I am new to caffe. Currently, I t...	4	If you got <code>caffe</code> from git you ...	7
2	28171577	2015-01-27 13:22:44	3066	<python><lua><caffe> <torch>	How to get a layer from a caffe model using torch	In python when I want to get the data from ...	2	First of all please note that <a href="http...	11
3	28177298	2015-01-27 18:17:48	30950	<python><caffe>	Import caffe error	i compiled caffe successfully in my ubuntu ...	14	This happens when you have not run <code>ma...	17
4	28692209	2015-02-24 09:33:33	7241	<python><caffe>	Using GPU despite setting CPU_Only, yielding u...	I'm installing Caffe on an Ubuntu 14.04 vir...	9	I'm gonna add a few words to Mailerdaimon's...	14
...	...	...	...	...	...	...	...	...	...

Рисунок 5.3 – Інформація дата фрейму

Для порівняння вхідних даних з питанням був використаний алгоритм Ретклiфф-Обершельпа, який був представлений в розділі 1.4. Даний метод реалізовано в бібліотеці difflib [36]. Створена функція `similarity`, яка приймає два аргументи, а саме тексти, та повертає рейтинг схожості (рис.5.4).

class difflib.SequenceMatcher - це гнучкий клас для порівняння пар послідовностей будь-якого типу, за умови, що елементи послідовності мають хеш. Об'єкт можна хешувати, якщо він має хеш-значення, яке ніколи не змінюється протягом усього життя, і його можна порівняти з іншими об'єктами.

ratio() – метод класу *difflib.SequenceMatcher*, який повертає міру схожості послідовностей в діапазоні від 0 до 1. Де 0 – немає схожості, а 1 – повна схожість.

```
difflib.SequenceMatcher(None, normalized1, normalized2).ratio()
```

Рисунок 5.4 – Реалізація порівняння двох текстів

5.3 Демонстрація роботи

Розроблена інформаційна система має інтерфейс, представлений на рисунку 5.5, який містить два елемента. Перший – це поле textarea у який потрібно ввести текст/програмний код для аналізу використання. Другий елемент – це кнопка пошуку, яка запускає функцію виконання моделі.

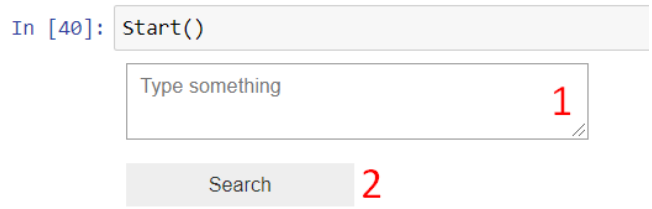

Рисунок 5.5 – Відображення інтерфейсу інформаційної системи

Розроблена система має перевірку на пусте поле введення(рис.5.6).

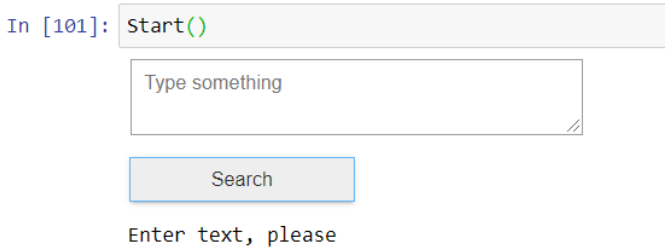

Рисунок 5.6 – Перевірка поля введення

Вставлений код/скрипт порівнюється з кожним «тілом» відповіді. Порівняння відбувається функцією `similarity`, який визначає рейтинг схожості двох текстів. Оскільки, змінивши порядок передачі аргументів, функція може повернути різні результати, було проведено два порівняння :

– першим аргументом функції передавались дані роботи, другим аргументом передавався текст питання запису із Stack Overflow;
– першим аргументом функції передавався текст питання запису із Stack Overflow, другим - дані роботи.

Порівняння записувались в окремому стовбці у дата фреймі.

Результатом роботи є запис, який має найбільшу схожість (сума двох порівнянь) з введеними даними. Якщо найбільша сума схожості менше значення 0.01 – то вважається, що результат не знайдений (рис.5.7).

Якщо результат знайдений, то він буде складатися з трьох частин. Приклад результату зображений на рисунку 5.8. Перший елемент – текст відповіді. Другий елемент – основна інформація про запитання із Stack Overflow. А саме: посилання на сайт з питанням, оцінка питання, оцінка відповіді, дата створення, кількість переглядів, теги запитання. Третій елемент – Заголовок і тіло питання.

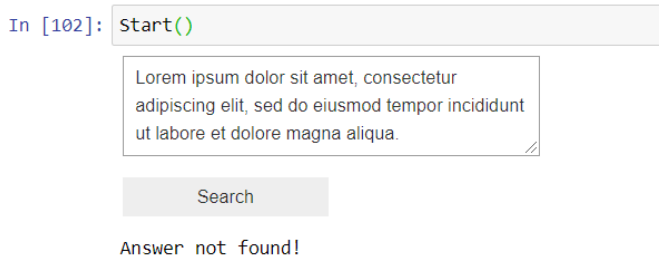

Рисунок 5.7 – Текст не знайденої відповіді

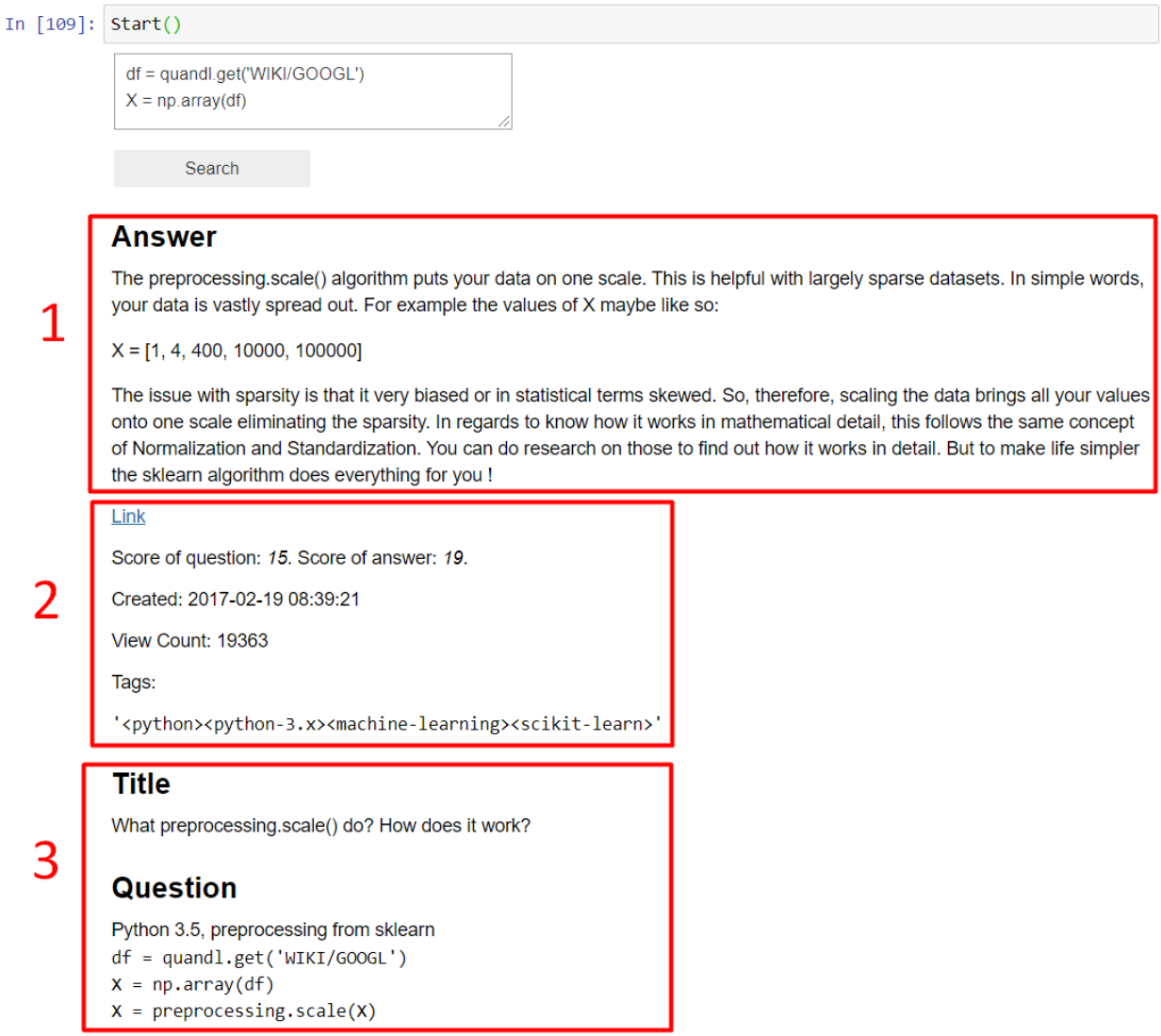

Рисунок 5.8 – Результат аналізу введених даних

ВИСНОВКИ

Зростаюче використання машинного навчання в розробці програмного забезпечення відкрило нові дослідницькі завдання для дослідників повторного пошуку. В роботі досліджується деякі проблеми, з якими стикаються розробники і які вимагають негайної уваги як з боку дослідників, так і з боку розробників API. Ключові висновки показують, що при розробці, швидше за все, зіткнуться з труднощами, в першу чергу, при підготовці даних, створення моделі і навчанні на етапах розробки машинного навчання. Всі ці висновки вказують на розробку нових принципів і парадигм для розробки API для машинного навчання і різних інструментів статичного аналізу.

Було проведено комплексне дослідження для розуміння закономірностей виникнення помилок при розробці програмного продукту. Було виявлено, що існують деякі унікальні шаблони помилок, такі як IPS, помилка потоку даних і помилка створення моделей, які відрізняються від інших областей розробки програмного продукту. Це означає, що потрібні нові види відладників і детекторів помилок, які допоможуть в розробці програмних продуктів.

Були знайдені деякі нові моделі виправлення, такі як виправлення підключення до мережі, виправлення розмірів даних, розмірів фіксуючого шару і т.д., які є новими і унікальними для розробки програмних продуктів. Шаблони фіксації можуть бути використані для автоматизації виправлення моделей.

Також було запропоновано автоматичну методику виявлення зловживань API, допущених розробниками.

Запропоновано інформаційну технологію аналізу результатів використання бібліотек машинного навчання, для її апробації розроблено інформаційну систему.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in Proceedings of the 22nd ACM international conference on Multimedia, pp. 675–678, ACM, 2015.
2. Candel, V. Parmar, E. LeDell, and A. Arora, “Deep learning with h2o,” H2O. ai Inc, 2016.
3. F. Chollet et al., “Keras.” https://github.com/fchollet/keras, 2015.
4. S. Owen and S. Owen, Mahout in action. Manning Shelter Island, NY, 2015.
5. X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al., “Mllib: Machine learning in apache spark,” The Journal of Machine Learning Research, vol. 17, no. 1, pp. 1235–1241, 2016
6. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., “Scikit-learn: Machine learning in python,” Journal of machine learning research, vol. 12, no. Oct, pp. 2825–2830, 2011.
7. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M.Isard,etal.,“Tensorflow: Asystemforlarge-scalemachinelearning.,”inOSDI,vol.16,pp.265– 283, 2016.
8. J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. WardeFarley, I. Goodfellow, A. Bergeron, et al., “Theano: Deep learning on gpus with python,” in NIPS 2011, BigLearning Workshop, Granada, Spain, vol. 3, Citeseer, 2011.
9. R. Collobert, S. Bengio, and J. Mariéthoz, “Torch: a modular machine learning software library,” tech. rep., Idiap, 2002.
10. G. Holmes, A. Donkin, and I. H. Witten, “Weka: A machine learning workbench,” in Intelligent
11. What is an API? (Application Programming Interface). [Електронний ресурс] - https://www.mulesoft.com/resources/api/what-is-an-api
12. SDLC - жизненный цикл разработки системы. [Електронний ресурс]- https://myalm.ru/news/SDLC
13. Machine Learning Engineer. https://jobs.lever.co/kite/1b9ea768-2192-4f4c-8521-3b484f0863ba

14. Codata — AI Pair Programmer. [Електронний ресурс] – https://medium.com/@pranavathiyani/codata-ai-pair-programmer-b1d6d7f8696
15. A.BaltadzhievaandG.Chrupała,“Predicting the quality of questions on stackoverflow,”in Proceedings of the International Conference Recent Advances in Natural Language Processing, pp. 32–40, 2015.
16. JON PECK, “Enrich data in Tableau with machine learning using”, 2019.
17. Mayank Gupta, “NUMPY in Python for Machine Learning”, 2020.
18. Steven Bird, Ewan Klein and Edward Loper, “*Natural Language Processing with Python*”, 2019.
19. TensorFlow, “TensorFlow Github,” 2019. https://github.com/tensorflow/tensorflow/tags/.
20. TensorBoard: инструментарий визуализации TensorFlow.2020. [Електронний ресурс] – https://www.tensorflow.org/tensorboard
21. Bird Steven, Klein Ewan, Loper Edward «Natural Language Processing with Python», 2009
22. Мера Жаккара: [Електронний ресурс] https://habr.com/ru/company/Voximplant/blog/446738/
23. Leskovec J., Rajaraman A., Ullman J.D. Mining of Massive Datasets (2nd ed.). New-York: Cambridge University Press, 2014.
24. Коэффициент Сёренсена: https://en.wikipedia.org/wiki/Sørensen–Dice_coefficient (дата обращения: 16.03.2020).
25. Ilya Ilyankou: Comparison of Jaro-Winkler and Ratcliff/Obershelp algorithms in spell check, May 2014
26. Documentation Python.org. [Електронний ресурс] - https://www.python.org/
27. JupyterLab и Jupyter Notebook — мощные инструменты Data Science [Електронний ресурс] – https://proglib.io/p/jupyter
28. Концептуальная модель. [Електронний ресурс] – https://studref.com/417921/sotsiologiya/kontseptualnaya_model
29. Знакомство с нотацией IDEF0 и пример использования. [Електронний ресурс] – https://habr.com/ru/company/trinion/blog/322832/

30. Machine Learning Engineer. [Електронний ресурс] – https://jobs.lever.co/kite/1b9ea768-2192-4f4c-8521-3b484f0863ba
31. S. Biswas, M. J. Islam, Y. Huang, and H. Rajan, “Boa meets python: A boa dataset of data science software in python language,” in MSR’19: 16th International Conference on Mining Software Repositories, May 2019.
32. R. Pan, M. J. Islam, S. Ahmed, and H. Rajan, “Identifying classes susceptible to adversarial attacks,” arXiv preprint arXiv:1905.13284, 2019
33. M. J. Islam, G. Nguyen, R. Pan, and H. Rajan, “A comprehensive study on deep learning bug characteristics,” in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2019, (New York, NY, USA), pp. 510–520, ACM, 2019.
34. About - Stack Exchange. [Електронний ресурс] – https://stackexchange.com/about
35. pandas documentation. [Електронний ресурс] – https://pandas.pydata.org/docs
36. difflib — Helpers for computing deltas. [Електронний ресурс] – https://docs.python.org/3/library/difflib.html

ДОДАТОК А

Планування робіт

Розбиття роботи на менші завдання - це загальний прийом продуктивності, який використовується для того, щоб зробити роботу більш керованою та доступною. Для проектів Структура розподілу робіт (WBS) є інструментом, який використовує цю техніку, і є одним з найважливіших документів управління проектами. Він одноосібно інтегрує базові показники обсягу, вартості та графіку, забезпечуючи узгодженість планів проектів.

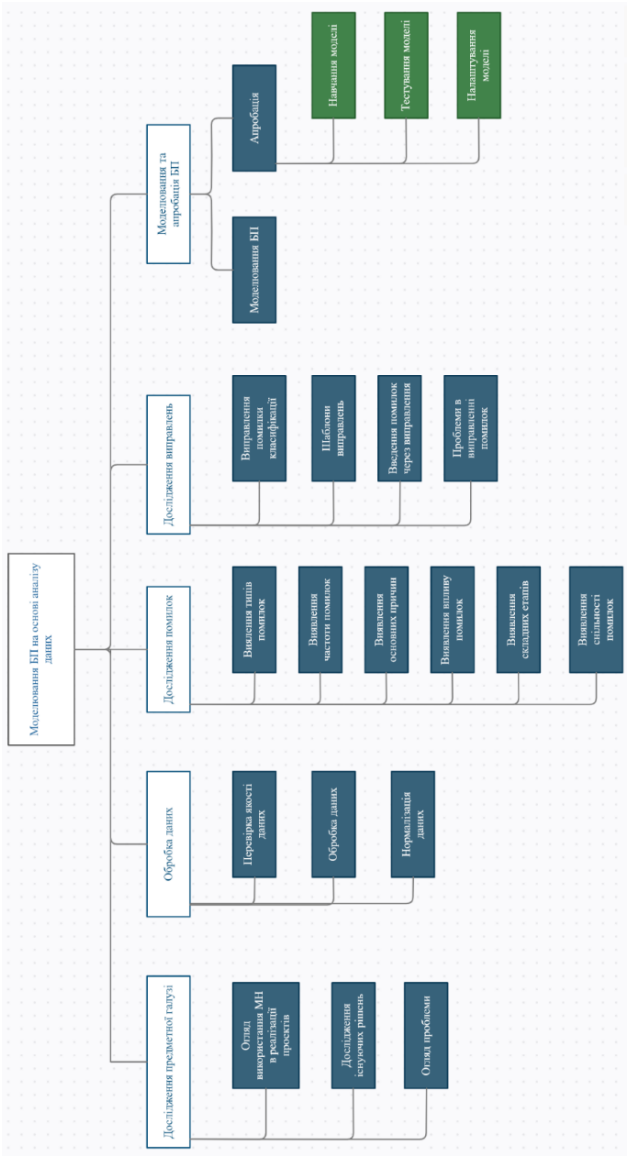

Рисунок А.1 – Таблиця WBS (work breakdown structure)

Побудова календарного графіку виконання ІТ – проекту

Для визначення часу здійснення заходів, спрямованих на досягнення цілей проекту, і для встановлення взаємозв'язків між ними по тимчасовому параметру з урахуванням найбільш ризикових подій, складається календарний план проекту.

В результаті створення календарного плану виходить повне проектне розклад, що враховує тривалість робіт і ресурсну базу, необхідну для виконання проекту.

Діграма Ганта представляється смугами, зорієнтовані вздовж осі (шкали) часу так, що кінець і початок кожної смуги відповідає часу початку і кінця роботи по виконанню завдання. Відповідно, довжина діаграми Ганта дозволяє визначити тривалість виробленої роботи. На інший (перпендикулярній) шкалою шикуються виконувані завдання. Стовпчики, що представляють ці завдання, взаємопов'язані між собою, а їх зв'язок відбивається фігурними стрілками.

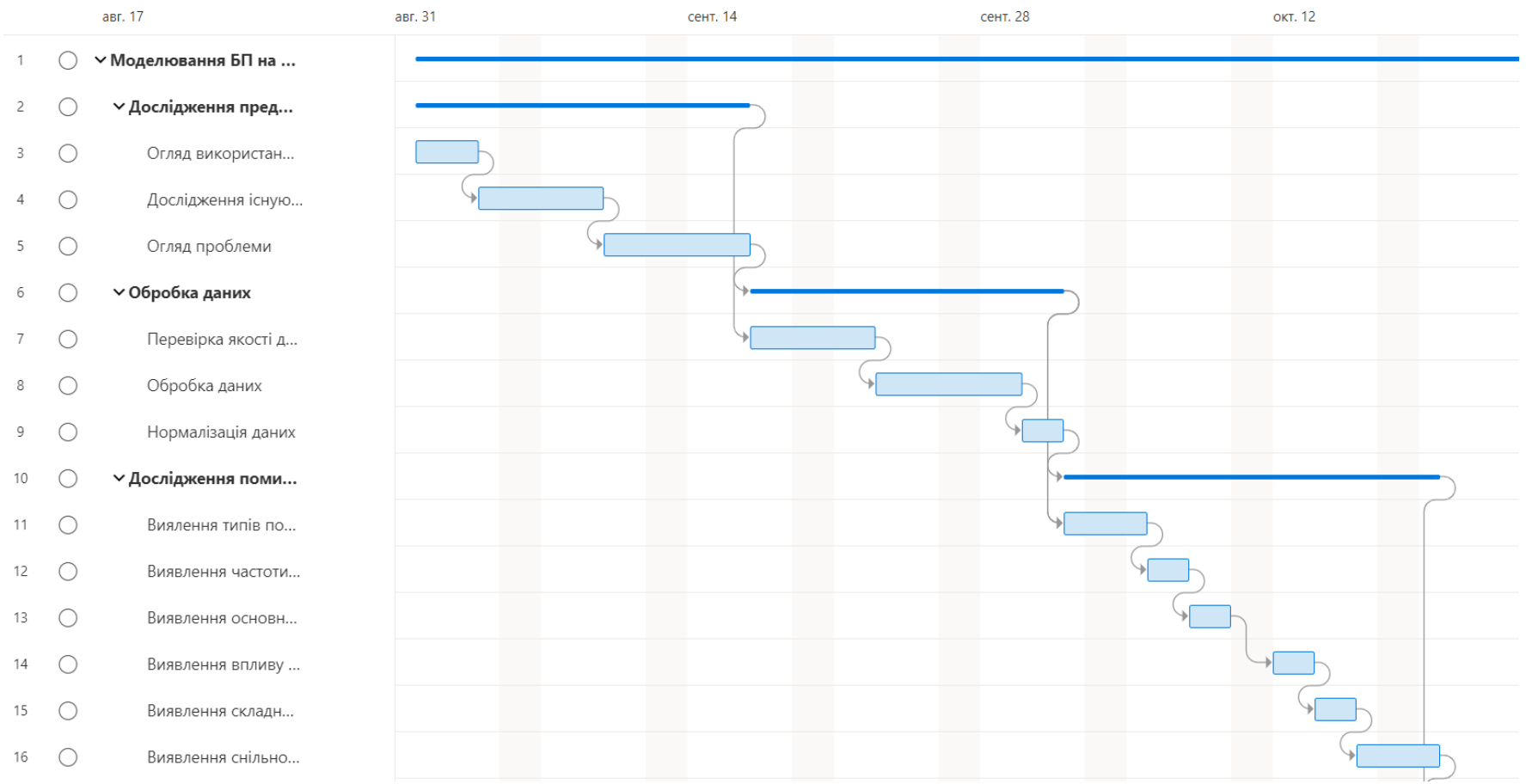

Рисунок А.2 – Діаграма Ганта

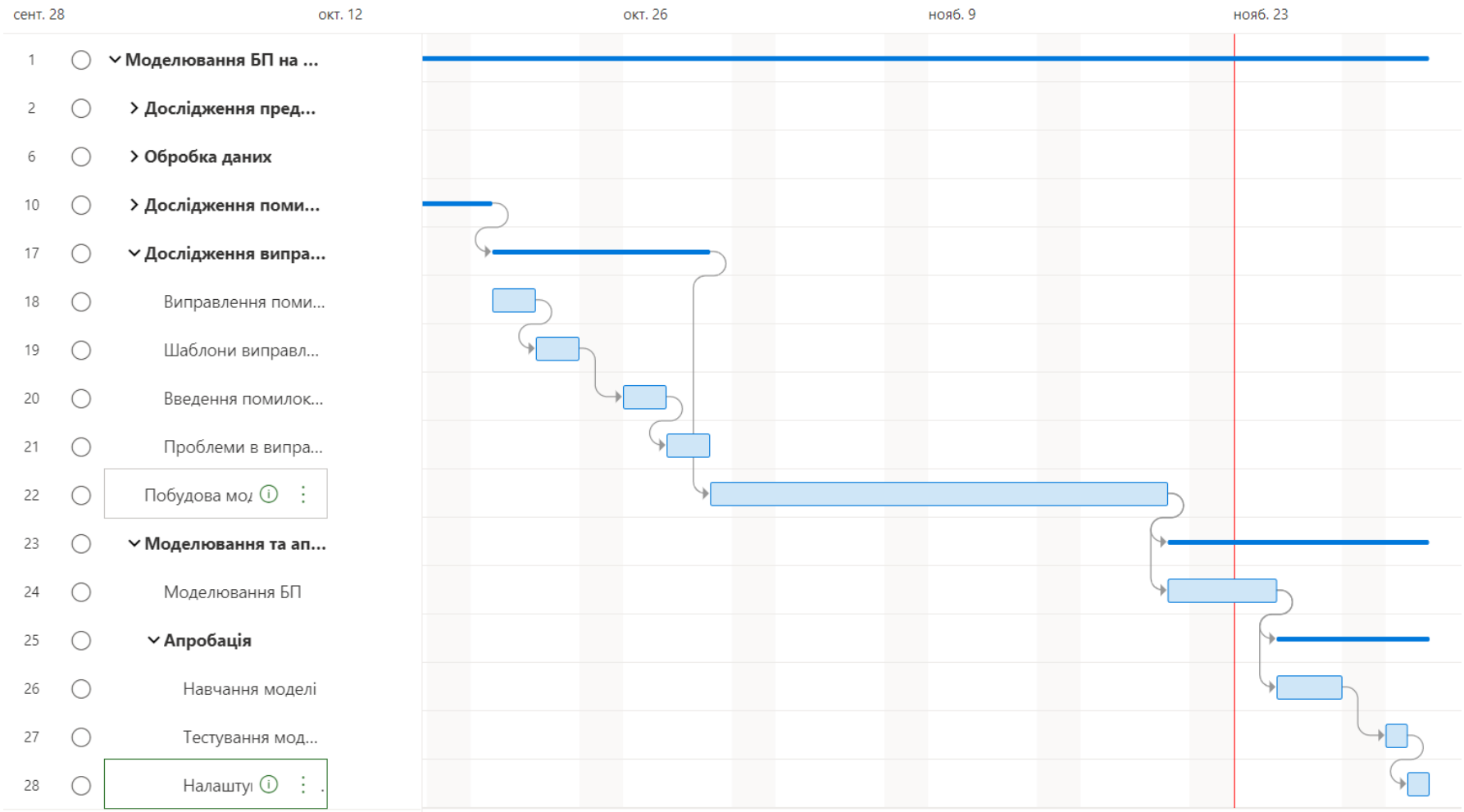

Рисунок А.3 – Продовження діаграми Ганта

Планування ризиків проекту

Деталізація мети проекту методом SMART. Результати деталізації методом SMART розміщені у таблиці А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити модель та інформаційну технологію аналізу використання бібліотек машинного навчання у реалізації проектів розробки програмних продуктів
Measurable (вимірювана)	Визначення помилок в реалізації проектів розробки програмних продуктів
Achievable (досяжна)	Автоматично визначати помилку при реалізації проектів розробки програмних продуктів з використанням алгоритмів машинного навчання
Relevant (реалістична)	Прискорити процес розробки програмних продуктів з використанням машинного навчання допомагаючи визначити помилки
Time-framed (обмежена у часі)	Реалізувати необхідний функціонал моделі не пізніше запланованого із замовником терміну.

Управління ризиками

До основних ризиків моделі та інформаційної технології аналізу результатів використання бібліотек машинного навчання у реалізації проектів розробки програмних продуктів відносяться:

− зміна цілей у ході реалізації проекту;

− зміна строків виконання роботи;

− не правильно оброблені дані;

− людський фактор;

− відсутність кваліфікованого програміста;

– зростання вимог до проекту;

Шкала оцінки ризику може відповідати емпіричній шкалі оцінки ризику(див табл. 2):

5 бали - критичний ризик (0,81 - 1);

4 бали - вагомий ризик (0,61 - 0,8);

3 бали – помірний ризик (0,41 - 0,6);

2 бали - незначний ризик (0,31 - 0,4);

1 бал - ігнорується ризик (0 - 0,3).

R = P * L, де R – рівень ризику; L – втрати, P – ймовірність виникнення.

Таблиця А.2 – Ризики проекту

№	Об’єкт ризику	P	L	R
1	Зміна цілей проекту	0,6	0,5	0,3
2	Зміна строків проекту	0,5	0,5	0,25
3	Неправильно оброблені дані	0,9	0,9	0,81
4	Людський фактор	0,4	0,3	0,12
5	Відсутність кваліфікованого робітника	0,5	0,6	0,3
6	Зростання вимог до моделі	0,3	0,3	0,09

ДОДАТОК Б

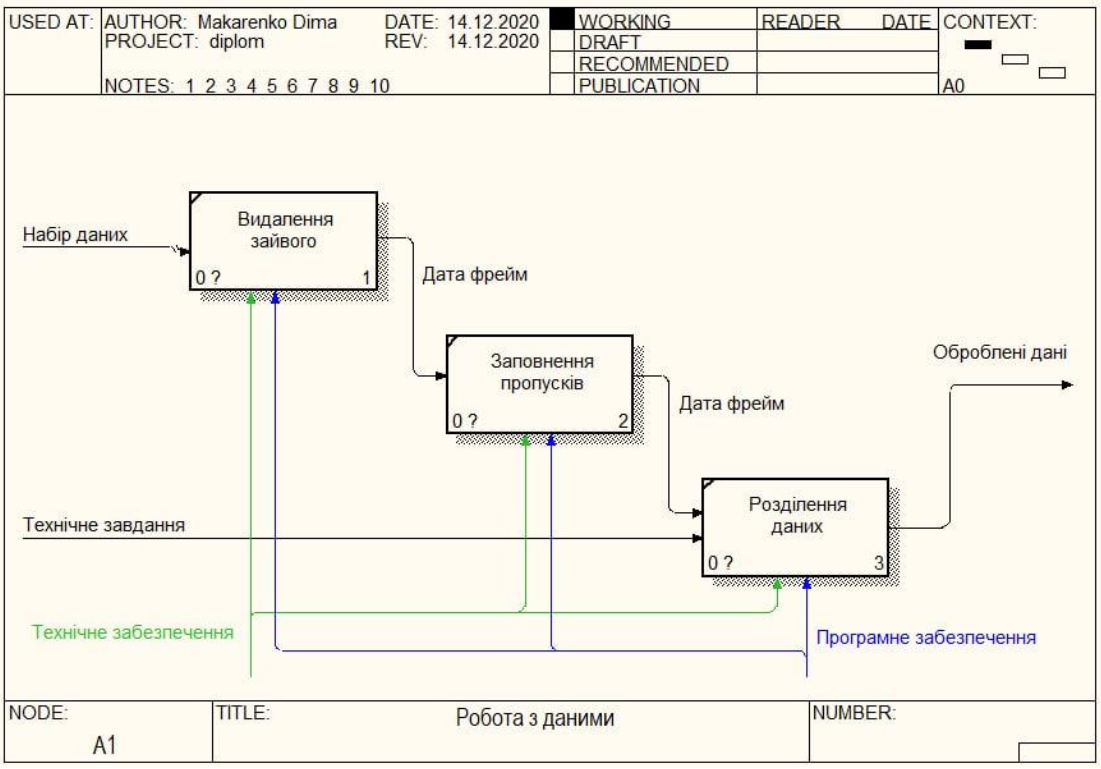

Рисунок Б.0.1 – Декомпозиція блоку «Робота з даними»

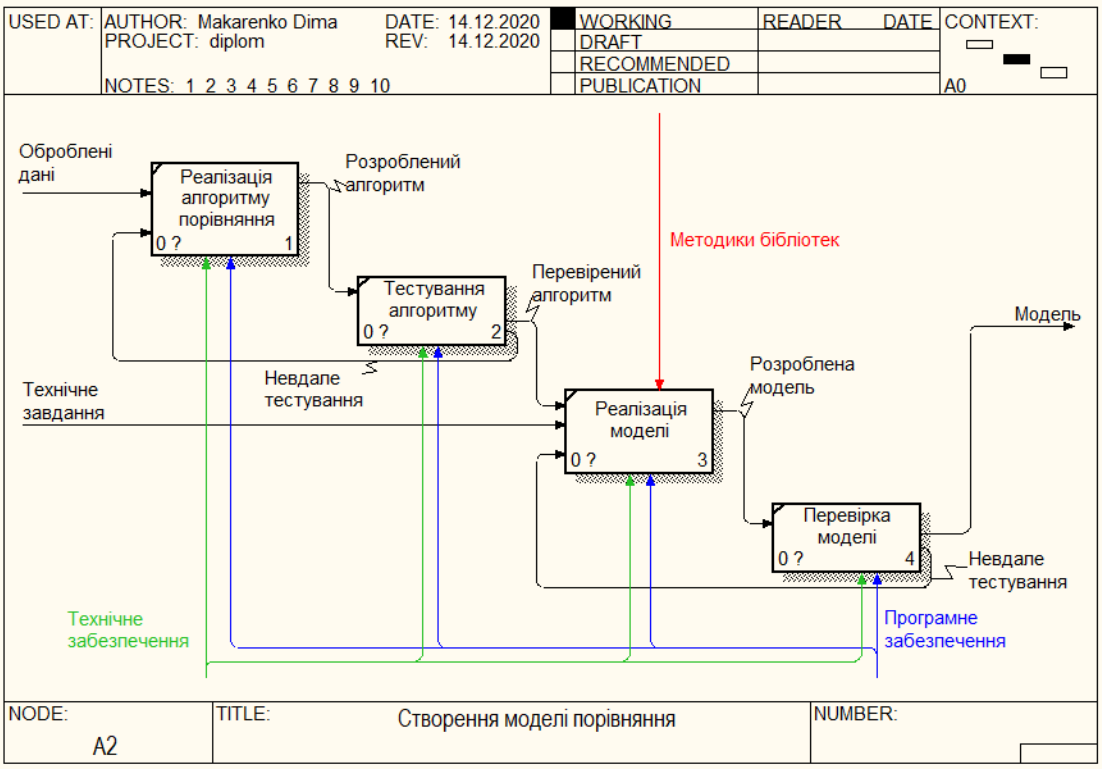

Рисунок Б.0.2 – Декомпозиція блоку «Створення моделі порівняння»

ДОДАТОК В

Код інформаційної моделі:

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import difflib
from IPython.core.display import display, HTML,clear_output
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
```

In [2]:

```
def similarity(s1, s2):
    normalized1 = s1.lower()
    normalized2 = s2.lower()
    return difflib.SequenceMatcher(None, normalized1, normalized2).rat
io()
```

In [3]:

```
#Caffe
df_Caffe = pd.read_csv("./data/Caffe.csv", nrows=184)
df_Caffe = df_Caffe.dropna()
#H2O
df_H2O = pd.read_csv("./data/H2O.csv", nrows=52)
df_H2O = df_H2O.dropna()
#Keras
df_Keras = pd.read_csv("./data/Keras.csv", nrows=1559)
df_Keras = df_Keras.dropna()
# Mahout
df_Mahout = pd.read_csv("./data/Mahout.csv", nrows=20)
df_Mahout = df_Mahout.dropna()
# Mllib
df_Mllib = pd.read_csv("./data/Mllib.csv", nrows=166)
df_Mllib = df_Mllib.dropna()
# Scikit-learn
df_Scikit_learn = pd.read_csv("./data/Scikit-learn.csv", nrows=9999999
)
df_Scikit_learn = df_Scikit_learn.dropna()
# Tensorflow
df_Tensorflow = pd.read_csv("./data/Tensorflow.csv", nrows=3083)
df_Tensorflow = df_Tensorflow.dropna()
# Theano
df_Theano = pd.read_csv("./data/Theano.csv", nrows=147)
df_Theano = df_Theano.dropna()
# Torch
df_Torch = pd.read_csv("./data/Torch.csv", nrows=95)
df_Torch = df_Torch.dropna()
# Weka
df_Wekka = pd.read_csv("./data/Weka.csv", nrows=62)
df_Wekka = df_Wekka.dropna()
```

In [4]:

```
df = pd.concat([df_Caffe,df_H2O, df_Keras,df_Mahout,df_Mllib,df_Scikit
_learn,
                df_Tensorflow,df_Theano,df_Torch,df_Wekka])
```

In [5]:

```
def search(s1, _df):
    for index, row  in _df.iterrows():
        _df.loc[index, 'Similarity'] = similarity(row['Body'], s1)
#         _df.loc[index, 'Similarity'] = 0

    return _df.sort_values(by='Similarity', ascending=False)
```

In [6]:

```
def Selector(x):
    sel.value = x
```

In [7]:

```
button = widgets.Button(description="Search")
output = widgets.Output()

Textarea = widgets.Textarea(
    value='',
    placeholder='Type something',
    description='',
    disabled=False
)
```

In [8]:

```
def displayA(ans):
    display(HTML('<h2>Answer</h2>'))
    display(HTML(ans['Body.1']))
    display(HTML('<a href='+ "https://stackoverflow.com/questions/" +
str(ans['Id']) + '>Link</a>'))
    display(HTML('<p>Score of question: <i>' + str(ans['Score']) +
                 '</i>. Score of answer: <i>' + str(ans['Score.1']) +
'</i>.</p>'))
    display(HTML('<p>Created: ' + str(ans['CreationDate']) + '</p>'))
    display(HTML('<p>View Count: ' + str(ans['ViewCount'])+ '</p>'))
    display(HTML('<span>Tags:' + ans['Tags']+ '</span>'))
    display(ans['Tags'])

    display(HTML('<h2>Title</h2>'))
    display(HTML(ans['Title']))
    display(HTML('<h2>Question</h2>'))
    display( HTML(ans['Body']))
```

In [9]:

```
def on_button_clicked(b):
    if Textarea.value == '':
        print('Enter text, please', end="\r")
    else:
```

```
        new_df = search(Textarea.value, df_Scikit_learn)

        if new_df.iloc[0]['Similarity'] == 0:
            print('Answer not found!')
        else:
            displayA(new_df.iloc[0])

button.on_click(on_button_clicked)
```

In [10]:

```
def Start():
    display(Textarea,button)
```

In [11]:

```
Start()
```

df = quandl.get('WIKI/GOOGL') X = np.array(df)